Mixture of Normalizing Flows for European Option Pricing

Yongxin Yang[1] **Timothy M. Hospedales**[2,3]

[1]Queen Mary University of London
[2]University of Edinburgh
[3]Samsung AI Centre, Cambridge

Abstract

We present a mixture of normalizing flows (MoNF) approach to European option pricing with guarantees that its estimations are free from static arbitrage. In contrast to many existing methods that meet economic rationality constraints (e.g., non-arbitrage) by introducing auxiliary losses, our solution meets those constraints exactly by design. To achieve this, we propose to build a model for risk neutral density using normalizing flows, which results in a pricing model, instead of modelling the option pricing function directly. First, we convert the constraints for direct pricing models to the constraints for models backed by risk neutral density estimation, then we design a specific NF architecture that meets these constraints. Furthermore, we find that employing a mixture of such normalizing flows improves the performance significantly, compared to using a deeper single NF. Finally, we present a mechanism to regularise the proposed model, and this regularisation can serve as a bridge between our method and any sample-based mathematical finance method. The evaluations on five option datasets show superiority of our method compared to mathematical finance solutions and some other neural networks based methods. The code is available at `https://github.com/qmfin/MoNF`.

1 INTRODUCTION

Option pricing has long been an active research area in both mathematical finance and machine learning community. Option pricing models provide a window to explain financial market mechanics, while efficient pricing models to set bid and ask prices in derivative markets are very valuable for practitioners. The seminal work Black–Scholes [Black and Scholes, 1973] initiated modern derivative theory by giving a theoretical estimate of European option price. Since then many studies have been carried out on finding better option pricing models in terms of fitting real market data more reliably.

The research paradigm of mathematical finance (MF) usually starts from a set of assumptions, e.g., what kind of stochastic process provides a sensible and accurate model of market movements. Based on these assumptions, they end up with a parametric model that takes as input some market signals (e.g., moneyness, time to maturity, and risk-free rate) then outputs the corresponding option price. The parameters in MF models usually have physical meaning in econometric theory, and thus they are more transparent and explainable, compared to their black-box machine learning counterparts. However, when a MF model fails to fit the market data, i.e., underfitting in machine learning sense, it is not straightforward to introduce more parameters.

On the other side, machine learning approaches treat option pricing as a regression problem. Though the input-output pairs are almost the same as the MF models above, researchers tend to use generic methods, such as kernel machines and neural networks [Malliaris and Salchenberger, 1993]. As a result, ML models can fit the market data nearly perfectly (if not, one can always choose to increase the model capacity). However, ML models usually fail important sanity checks, e.g., they would give nonsense predictions for extreme cases where no training data are available. This makes them less appealing to practitioners since their behaviours can be unpredictable for the rare events.

There are some attempts to integrate the econometric axioms into a machine learning model as inductive bias. More specifically, there are several conditions for an economically rational pricing model, and some studies [Dugas et al., 2000, Yang et al., 2017, Ackerer et al., 2020] try to meet these conditions by architecture design, auxiliary losses, and data augmentation. However, those methods can only meet the part of conditions with guarantees, and the rest of them are dealt with penalties that can be violated.

Our insight is that meeting all conditions for a pricing model is intrinsically hard, but it becomes much easier if we choose to work on the risk neural density (the probability density of the underlying asset's price on the maturity date) [Bahra, 1997], which comes with the corresponding conditions.

In this work, we propose a normalizing flows based model for risk neutral density, which results in a deterministic pricing function in the end. By specific architecture and activation function design, our method can meet all financial axiom conditions, and yields very accurate estimates. Furthermore, our approach sheds light on the less studied issue of how to bridge mathematical finance and machine learning approaches to option pricing.

2 RELATED WORK

2.1 OPTION PRICING

Option pricing is an active research area in mathematical finance. The seminal work Black–Scholes [Black and Scholes, 1973] assumed geometric Brownian motion (GBM) for the underlying asset price and provided a closed-form pricing formula with only one free parameter – volatility (the standard deviation of logarithmic returns). The GBM assumption is usually invalidated in real market, and there are two mainstream research directions for a fix. The first direction introduces randomness to the volatility, rather than treating it as a constant. For example, [Heston, 1993] allowed arbitrary correlation between volatility and asset returns. It introduced stochastic interest rates as well, and demonstrated the effectiveness for bond options and foreign currency options. [Barndorff-Nielsen, 1997] took the normal inverse Gaussian law as a building element, exploring the construction of analytically and statistically tractable stochastic processes. [Madan et al., 1998] used the Variance Gamma process (a.k.a. the Laplace motion) because it allows for a wider of skewness and kurtosis than the Brownian motion. The second direction is to add a jump process, which represents rare events, in addition to GBM. [Merton, 1976] superimposed a jump component on a diffusion component. The jump component is composed of log-normal jumps driven by a Poisson process. [Carr and Geman, 2002] generalised the Variance Gamma model [Madan et al., 1998] to allow for both diffusion process and jump process. [Kou, 2002] proposed a double exponential jump-diffusion model, incorporate the leptokurtic feature that the return distribution of assets having a higher peak and two heavier tails than normal distribution, and strike a balance between reality and tractability. A comprehensive study over those models can be found in [Jeanblanc et al., 2009].

Apart from working on the pricing formula directly, it is possible to build the model through other proxies. Possible indirect modelling approaches are: (i) Implied volatility [Ackerer et al., 2020], which is essentially the inverse function of Black–Scholes formula; One can build the model for implied volatility curve (SVI) or implied volatility surface (SSVI) and then apply Black–Scholes formula for pricing. (ii) Risk neural density [Monteiro et al., 2008], which models the probability distribution of asset prices in future. The majority of mathematical finance methods model the stochastic process of how asset price evolves, which naturally lead to density functions along time axis. However, they may not have analytic forms for these density functions, therefore the pricing has to be done by Monte Carlo.

2.2 NEURAL NETWORKS FOR OPTION PRICING

The use of neural networks for option pricing can be traced back to 1990s [Malliaris and Salchenberger, 1993, Hutchinson et al., 1994]. For a comprehensive review, please refer to [Ruf and Wang, 2020]. At the first glance, neural networks appear to be a good fit because they are universal approximators, and it is easy to increase the number of parameters in the case of under-fitting. In contrast, the parameters in white-box mathematical finance models usually have physical meanings, and it is non-trivial to extend those models by introducing more parameters. However, option pricing is not simply mapping the input features (e.g., strike price, time to maturity) to the market prices. Rational option pricing models have to meet a series of conditions to guarantee that they rule out any arbitrage opportunities. To this end, many neural networks models borrow the insight from the mathematical finance models. To name a few, [Garcia and Gençay, 2000] designed a NN-based option pricing model with a similar form of Black–Scholes formula. [Dugas et al., 2000] chose specific activation functions and weight constraints so that their model has the same properties on first-order and second-order derivatives as the mathematical finance models. [Yang et al., 2017] adapted these strategies and also utilised *learning from hint* [Abu-Mostafa, 1990] for the conditions that are hard to meet with architecture design. [Ackerer et al., 2020] extended those techniques to implied volatility surface fitting and demonstrated their effectiveness. These studies showed that employing constraints required for econometric rationality produces better option pricing models compared to vanilla unconstrained feed-forward neural networks.

Overall, neural networks are promising methods for option pricing, and there exist some successful studies in which rationality constraints are exploited. However, they usually use auxiliary loss functions to enforce the constraints, as a result, those constraints are not guaranteed to hold at all times. The reason for using auxiliary losses rather than exploiting architecture design is that the constraints are in the form of taking the limit or the higher-order gradient w.r.t. the input of the neural network. So they are hard to meet as the neural networks go deeper. For example, [Dugas et al., 2000, Yang et al., 2017] have only one hidden layer, and the techniques developed can not be extended beyond that. In our work, we choose to model the risk neutral density (RND), and derive the corresponding constraints for it. We find that it becomes much easier to deal with these constraints in RND space.

2.3 NORMALIZING FLOWS

The key ingredient of the proposed method is called normalizing flows (NF), which is a popular generative model in machine learning [Papamakarios et al., 2019]. The unique advantage of NF over other generative models is that it allows exact likelihood computation, which leads to a closed-form pricing formula in our case. The basic idea of NF is to stack a number of invertible layers, such as coupling [Dinh et al., 2015, 2017], autoregressive [Kingma et al., 2016], masked autoencoder [Papamakarios et al., 2017], and invertible convolution [Kingma and Dhariwal, 2018], such that the data distribution is transformed to a pre-defined distribution (e.g., Gaussian). Studies in this area usually focus on designing an invertible operator with small computation cost of the determinant (e.g., the operator produces a triangular matrix). Recently, NFs have been extended to the continuous domain [Grathwohl et al., 2019]. Their applications beyond generating realistic data samples are also exploited, e.g., in lossless compression [Hoogeboom et al., 2019]. An early attempt of using NFs for RND modelling was [Loaiza-Ganem et al., 2017], and it followed the principle of maximum entropy from the previous studies [Buchen and Kelly, 1996, Neri and Schneider, 2012]. However, it is effectively not risk-neutral because the constraint over expectation of asset's future price is realised by an auxiliary loss, which is not guaranteed to be zero.

3 METHODOLOGY

3.1 PROBLEM SETTING

A European option is a contract that gives the holder the right, but not the obligation, to buy (call option) or to sell (put option) the underlying asset (e.g., stock) at a specified price (strike price) on a certain future date (maturity date). For example, at time $t = 0$ (i.e., today), a company's stock price is \$100, and an individual buys a call option with strike price \$110 and maturity date $T = 5$ (five days later). After five days, if the company's stock price is \$120, he can exercise the option and get the stock at the strike price \$110. In this case, if he sells the stock immediately, he will get \$10 profit. One the other hand, if the company's stock price is below \$110, he will choose not to exercise the option, and the only loss for him is the price of the option (sometimes called premium). For the put option, he profits if the stock price is lower than the strike price on the maturity date, as he can buy the stock from the market at a lower price and sell it at the strike price.

The problem setting of option pricing is to answer: what should the price for the option at $t = 0$ be? In this work, we restrict ourselves to the case of modelling option price curves, i.e., for a given (annualised) time to maturity $\tau = \frac{T-t}{365}$ and the current underlying asset price S_0, we seek for the "best" mapping from the strike price K to the option price y. For a parametric model f_θ, the process of fitting market data of N option prices (sometimes referred to as *calibration*) usually involves the following optimisation,

$$\min_\theta \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(K_i; z_i) - y_i \|_2^2 \qquad (1)$$

where K_i refers to the i-th option's strike price; z_i refers to other factors such as the asset price S_0, time to maturity τ, type of option (put/call), risk-free rate, and dividend yield of the option's underlying asset (if any); y_i refers to the i-th option's price. The optimised model can be used to price the option contract with unseen strike prices.

3.2 THE CONSTRAINTS FOR NON-ARBITRAGE

At the first glance, Eq. 1 appears to be a univariate regression problem. However, f_θ has to meet a series of constraints such that it rules out arbitrage opportunities.

More specifically, we consider the static arbitrage for pricing curves defined in [Carr et al., 2003], which include the following constraints for call options

$$\frac{\partial f_\theta(K)}{\partial K} \leq 0 \ \textbf{(C1)} \quad \frac{\partial^2 f_\theta(K)}{\partial K^2} \geq 0 \ \textbf{(C2)} \quad f_\theta(\infty) = 0 \ \textbf{(C3)}$$

$$\max(0, e^{-r\tau}(F_\tau - K)) \leq f_\theta(K) \leq e^{-r\tau} F_\tau \ \textbf{(C4)}$$

In (C4), the forward price F_τ is defined as $F_\tau = e^{(r-q)\tau} S_0$, where r is the continuously compounded zero-coupon interest rate and q is the continuously paid dividend yield. Note that, r and q have their own structures which need to be estimated separately in real-world trading, but it is out of the scope of this work. We treat them as known constants for a given time period τ in this work. For put options, we have similar constraints,

$$\frac{\partial f_\theta(K)}{\partial K} \geq 0 \ \textbf{(P1)} \quad \frac{\partial^2 f_\theta(K)}{\partial K^2} \geq 0 \ \textbf{(P2)} \quad f_\theta(0) = 0 \ \textbf{(P3)}$$

$$\max(0, e^{-r\tau}(K - F_\tau)) \leq f_\theta(K) \leq e^{-r\tau} K \ \textbf{(P4)}$$

The key insight of our work is that it is hard for a parametric model (e.g., neural networks) to meet all constraints above, but it is much easier if we chose to model the risk neural density instead.

We propose to model the density of forward log-moneyness of the asset price S_T on the maturity date, i.e., $x = \log(\frac{S_T}{F_\tau})$, because the common range of forward log-moneyness is $[-3, 3]$, which is more numerically stable than the asset price itself.

More specific, we design the following option pricing model

$$f_\theta(K; z) = e^{-r\tau} F_\tau \int_{-\infty}^{+\infty} [\mathbb{I}(z)(e^x - \frac{K}{F_\tau})]_+ q_\theta(x) \mathrm{d}x \quad (2)$$

where $[\cdot]_+ := \max(0, \cdot)$ and $\mathbb{I}(z)$ is an indicator function that returns $+1$ for call option and -1 for put option. Here we slightly abuse the notation z for the triplet (Put/Call, $F_\tau, e^{-r\tau}$) while the indicator function only applies to its first element – option type.

With Eq. 2, all constraints for pricing model are reduced to the following three constraints on the density estimator $q_\theta(x)$. (Proof can be found in the appendix.)

$$q_\theta(x) \geq 0 \;\textbf{(R1)} \qquad \int_{-\infty}^{+\infty} q_\theta(x) = 1 \;\textbf{(R2)}$$

$$\int_{-\infty}^{+\infty} e^x q_\theta(x) \mathrm{d}x = 1 \;\textbf{(R3)}$$

(R1) and (R2) are constraints for any valid density function, and (R3) sets the expectation of e^x to be constant 1, which means the expectation of underlying asset price at maturity date, $\mathbb{E}_q[S_T]$, should be the same as the forward price F_τ, and its discounted value (by risk-free rate r) at $t = 0$ would be $e^{-q\tau} S_0$, which corresponds to the upper bound of call option price in (C4). The density that meets (R3) is referred to as *risk-neutral* density, as it basically states that the expectation of an asset priced at S_0 will be $S_0 e^{(r-q)\tau}$ after the period of τ (or $S_0 e^{r\tau}$ if it does not pay any dividends).

3.3 A NORMALIZING FLOWS MODEL

In this work, we build $q_\theta(x)$ by a normalizing flows model, which maps a one-dimension standard Gaussian to the distribution of interest. More specifically, we have a sample $\epsilon \sim \mathcal{N}(0, 1)$. After a sequence of invertible transforms, it becomes a sample from the risk neural density, i.e., $\epsilon \xrightarrow{g_\theta} x$ where $g_\theta(\cdot)$ stands for the sequence of transforms.

To find the best θ, we can estimate Eq. 2 by Monte Carlo,

$$f_\theta(K; z) = e^{-r\tau} F_\tau \frac{1}{M} \sum_{m=1}^{M} [\mathbb{I}(z)(\exp(g_\theta(\epsilon_m)) - \frac{K}{F_\tau})]_+ \quad (3)$$

where $\{\epsilon_1, \epsilon_2, \ldots, \epsilon_M\}$ are samples from standard Gaussian. Then we can minimise Eq. 1 with Eq. 3 plugged-in by any gradient based method, as long as $g_\theta(\cdot)$ is differentiable.

$$\nabla_\theta \sum_{i=1}^{N} \| e^{-r\tau} F_\tau \frac{1}{M} \sum_{m=1}^{M} [\mathbb{I}(z_i)(\exp(g_\theta(\epsilon_m)) - \frac{K_i}{F_\tau})]_+ - y_i \|_2^2 \quad (4)$$

One may question the necessity of normalizing flows, because any generator appears to do the job in Eq. 4. The very unique advantage of normalizing flows is that we can get the exact density of $q(x)$ by

$$q_\theta(x) = \mathcal{N}(g_\theta^{-1}(x); 0, 1) \left| \frac{\mathrm{d}g_\theta^{-1}(x)}{\mathrm{d}x} \right| \quad (5)$$

Therefore we can run a numerical integration instead of Monte Carlo for Eq. 2 in the testing stage, such that we will have a *deterministic* result. This is crucial for real-world deployed applications, though we still prefer to do Monte Carlo during training for better efficiency. Note that, the interval of integration in Eq. 2 is $[-\infty, +\infty]$ theoretically, but $[-3, 3]$ is sufficient in practice, e.g., 3 means the underlying asset price increases by $\mathrm{e}^3 \approx 20$ times.

3.3.1 Enforcing the expectation constraint

It is non-trivial to meet (R3) by the architecture design, but it becomes possible for normalizing flows as they model the density explicitly.

If we have the following expectation calculated by numerical integration $\mu_\theta = \int_{-\infty}^{+\infty} e^x q_\theta(x) \mathrm{d}x$, it is easy to see $\int_{-\infty}^{+\infty} \frac{e^x}{\mu_\theta} q_\theta(x) \mathrm{d}x \equiv 1$. Therefore the following pricing model meets (R3) up to machine precision.

$$f_\theta(K; z) = e^{-r\tau} F_\tau \int_{-\infty}^{+\infty} [\mathbb{I}(z)(\frac{e^x}{\mu_\theta} - \frac{K}{F_\tau})]_+ q_\theta(x) \mathrm{d}x \quad (6)$$

However, differentiating through μ_θ (the numerical integrator) is unnecessarily expensive, thus we place the stop gradient operator $\mathrm{SG}(\cdot)$ over μ_θ, i.e.,

$$f_\theta(K; z) = e^{-r\tau} F_\tau \int_{-\infty}^{+\infty} [\mathbb{I}(z)(\frac{e^x}{\mathrm{SG}(\mu_\theta)} - \frac{K}{F_\tau})]_+ q_\theta(x) \mathrm{d}x \quad (7)$$

To understand the effect of $\mathrm{SG}(\mu_\theta)$, an equivalent realisation is: (i) duplicate θ into online parameter θ_1 and offline parameter θ_2 and (ii) use the following pricing formula,

$$f_{\theta_1, \theta_2}(K; z) = e^{-r\tau} F_\tau \int_{-\infty}^{+\infty} [\mathbb{I}(z)(\frac{e^x}{\mu_{\theta_2}} - \frac{K}{F_\tau})]_+ q_{\theta_1}(x) \mathrm{d}x \quad (8)$$

For optimisation, we initialise θ_1 and set $\theta_2 = \theta_1$ to ensure the expectation constraint (R3) is met, then alternate

Step (1) Update θ_1 by running one-step gradient descent for reducing the pricing errors.

Step (2) Update θ_2 by setting $\theta_2 = \theta_1$.

We can see that, Step (1) improves the fitness of pricing model to market data, but it breaks the expectation constraint. Step (2) fixes the expectation constraint issue, but it worsens the fitness. By alternating these two steps until convergence, we should have the final model that fits the

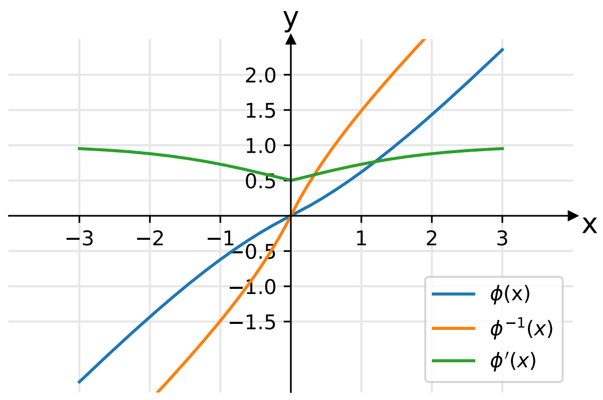


Figure 1: Plots of the proposed activation function (Eq. 10), its inverse, and its first order derivative.

market data well and meets the the expectation constraint. When it comes to the *actual* implementation, it is easier to use stop gradient $\mathrm{SG}(\mu_\theta)$ in Eq. 7 instead of the explicit alternating optimisation explained above. With Eq. 7, we modify the Monte Carlo estimator in Eq. 3 correspondingly.

$$f_\theta(K;z) = e^{-r\tau}F_\tau \frac{1}{M}\sum_{m=1}^{M}[\mathbb{I}(z)(\frac{\exp(g_\theta(\epsilon_m))}{\mathrm{SG}(\mu_\theta)} - \frac{K}{F_\tau})]_+$$

3.3.2 Activation function design

As we only need to transform a one-dimensional distribution to another, a layer in our model is simply,

$$h^{(L+1)} = \phi(a^{(L)} \times h^{(L)} + b^{(L)}) \tag{9}$$

where $\phi(\cdot)$ is an invertible activation function. Apart from being invertible, we introduce two more desired properties: (i) symmetric: we start from a standard Gaussian, which is symmetric, and we do not have bias towards positive or negative sides, i.e., we do not assume the asset price should increase or decrease. (ii) it is close to an identity function for the input with sufficiently large absolute value, because we need to avoid a possible explosion in values, in both $\epsilon \xrightarrow{f_\theta} x$ and $x \xrightarrow{f_\theta^{-1}} \epsilon$, given that multiple transformations (layers) can be stacked.

Frustratingly, the identity function $\phi(x) = x$ meets all the conditions above, but we want a degree of non-linearity, otherwise the model is reduced to Gaussian. To this end, we design the following activation function, based on some translation and rotation of the softplus function,

$$\phi(x) = \begin{cases} \log(1+\mathrm{e}^{x}) - \log(2) & \text{if } x \geq 0 \\ -\log(1+\mathrm{e}^{-x}) + \log(2) & \text{if } x < 0 \end{cases} \tag{10}$$

The plots of $\phi(x)$, its inverse $\phi^{-1}(x)$, and its gradient $\phi'(x)$ can be found in Fig. 1.

3.3.3 Mixture of normalizing flows

The model that we have developed so far has one drawback: it is hard to scale up by introducing more trainable parameters. Every layer contains two parameters only, i.e., the shift term a and the bias term b. To add more parameters in the case of under-fitting, we have to stack more layers. However, this would cause difficulties for training, e.g. we may have the gradient vanishing-exploding problem.

To add one more axis in the parameter space, we propose to use a mixture of normalizing flows models, rather than a single one, i.e.,

$$q(x) = \sum_{i=1}^{K}\pi_i q_i(x) \quad \text{s.t.} \quad \pi_i \geq 0, \sum_i \pi_i = 1 \tag{11}$$

Behind each q_i is an independent NF model, realised by $z \xrightarrow{f_{\theta_i}} x$.We can also verify that all conditions are still met as long as π is a probability simplex. In fact, the mixture of normalizing flows effectively creates K pricing models, and the final price is the re-weighted sum of the outputs of those models, i.e.,

$$\hat{c} = \mathrm{SOFTMAX}(\tilde{\pi}) \cdot [\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_K] \tag{12}$$

where $\tilde{\pi} \in \mathbb{R}^K$ and each $\hat{c}_i$ is estimated by Monte Carlo (Eq. 3). With Eq. 12, Eq. 3, and Eq. 1, we can train all parameters $\{\tilde{\pi}, \theta_1, \theta_2, \ldots, \theta_K\}$ using gradient based methods.

3.3.4 Regularisation

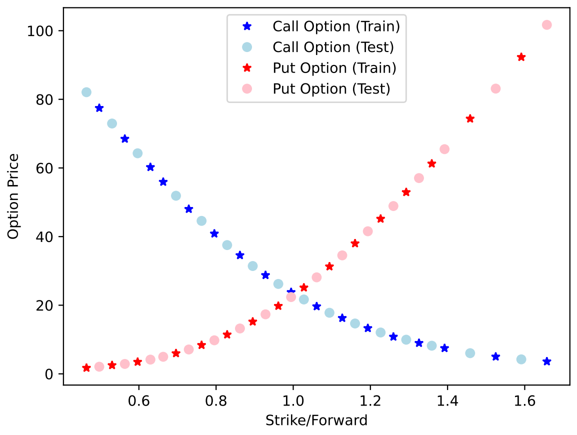
The mixture of normalizing flows can fit market data nearly perfectly, but it may result in an overly complicated density function (see zig-zag in Fig. 4). Though a non-smooth density function will not lead to a non-smooth pricing function, because the density function is just the second order derivative of pricing function, we want a mechanism for regularisation to reduce the risk of over-fitting[1].

Assume that we have a reference risk neutral density $\bar{q}(x)$, which could come from a well-calibrated mathematical finance model, or other model with much fewer parameters. To regularise our model $q_\theta(x)$, we can minimise the divergence of $q_\theta(x)$ and $\bar{q}(x)$, i.e.,

$$\min_\theta \mathcal{D}(q_\theta(x)||\bar{q}(x)) \tag{13}$$

Here we choose $\mathcal{D}(\cdot,\cdot)$ to be the Wasserstein distance (a.k.a. earth mover's distance), because it has a closed-form solution for sample-based approximation under the

[1]Strictly speaking, only non-smooth pricing function raise concerns over over-fitting. Hence the regularisation is mainly for elegance in machine learning rather than generalization.



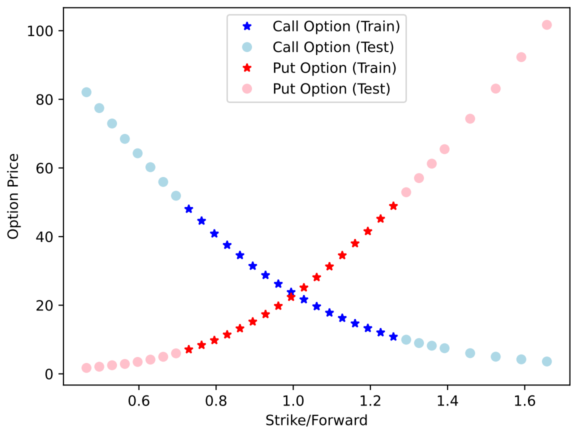


Figure 2: The two ways to split the training and testing set: Interpolation (left) and Extrapolation (right)

	Number of Contracts					Average Prices				
	Alphabet	Apple	Microsoft	NASDAQ100	S&P500	Alphabet	Apple	Microsoft	NASDAQ100	S&P500
Call	1,939,066	809,333	63,0234	4,292,133	10,029,359	253.69	35.75	30.29	1131.10	441.96
Put	1,943,777	803,095	63,4802	4,186,962	10,492,160	96.24	23.13	16.72	458.22	95.50

Table 1: Some Statistics of the Dataset: Number of Contracts (Left) and Average Prices (Right)

one-dimensional case. For example, if we draw M samples $\{\bar{x}_1, \bar{x}_2, \dots, \bar{x}_M\}$ from $\bar{q}(x)$, sort them in ascending order, i.e., $\bar{x}_{[1]} < \bar{x}_{[2]} < \cdots < \bar{x}_{[M]}$. Similarly, we get the sorted samples from $q_\theta(x)$, say, $x_{[1]} < x_{[2]} < \cdots < x_{[M]}$, then the estimate of Eq. 13 is,

$$\min_\theta \frac{1}{M} \sum_{i=1}^{M} |x_{[i]} - \bar{x}_{[i]}| \tag{14}$$

To calculate the gradient for Eq. 14, we need to differentiate a random sample x w.r.t. distribution parameters θ (in this case $\theta = \{\tilde{\pi}, \theta_1, \theta_2, \dots, \theta_K\}$). Since $q_\theta(x)$ is one-dimensional mixture of distributions, this can be done by the implicit gradient technique developed in [Graves, 2016, Figurnov et al., 2018].

$$\frac{\partial x}{\partial \theta} = -\frac{1}{q_\theta(x)} \frac{\partial}{\partial \theta} \int_{-\infty}^{x} q_\theta(u) \mathrm{d}u \tag{15}$$

The integral part in Eq. 15 can be estimated by numerical integration, starting from a sufficiently small value, i.e., -3, and ending by a value which is sufficiently close to x.

The techniques developed here can serve as a bridge between our machine learning model and mathematical finance models, enabling us to use a well-calibrated (well-trained) mathematical finance model to *regularise* our model, so that it behaves closely to the MF model. This is potentially useful to improve out-of-distribution predictions of our model.

Besides, many MF models do not have a risk neutral density or a pricing function in an analytic form, as they are essentially sample based. Therefore, we can approximate a MF model and obtain an analytic expression for RND using Eq. 14, which can be used for further analysis or simply a means for producing a deterministic price in a more efficient manner (numerical integration v.s. Monte Carlo).

4 EXPERIMENTS

We evaluate our method as well as several baselines on five option datasets including three stock options: Alphabet, Apple Inc, and Microsoft; two index options: NASDAQ100 and S&P500. Since there are not real ground truths for option prices, we assume that the frequently traded options reflect the true value of option contracts.

4.1 DATA PRE-PROCESSING

The full options dataset is provided by a third party, and we release an anonymised snapshot of data with the coverage over the period from 01-Jan-2017 to 31-Dec-2021[2].

The option price is the mid-point of best bid and best offer quotes by the end of day. Besides, we apply three filters: (i) we exclude all options with time to maturity shorter than 7 days or longer than 2 years (ii) we discard options with implied volatility larger than 1.5 as they are less frequently traded. (iii) we discard options with best offer smaller than 0.05 for the same reason. In total we have 35 million option contracts left after filtering. Some statistics on the number

[2]`https://github.com/qmfin/option_data`

	Alphabet		Apple		Microsoft		NASDAQ100		S&P500	
	Test	Train	Test	Train	Test	Train	Test	Train	Test	Train
Black Scholes	0.3436	0.3448	0.2284	0.2240	0.2149	0.2175	0.3156	0.3189	0.5354	0.5358
Heston	0.2855	0.2872	0.1825	0.1810	0.1185	0.1154	1.2811	1.3029	0.8356	0.8377
Kou Jump	1.0465	0.9165	0.6261	0.6589	0.4294	0.4246	1.3204	1.2927	1.7156	1.7563
Merton Jump	0.3384	0.3385	0.2284	0.2240	0.2149	0.2175	0.4049	0.4055	0.3794	0.3797
Variance Gamma	0.2351	0.2365	0.1150	0.1106	1.0360	1.3023	0.1324	0.1314	0.1494	0.1499
MNN	0.3427	0.3456	0.2325	0.2586	0.2275	0.2113	0.4221	0.4824	0.3883	0.3825
GNN	0.2896	0.3021	0.1921	0.2242	0.1300	0.1147	1.0503	1.2572	0.7849	0.8472
MoNF (Linear)	0.4791	0.5077	0.1209	0.1325	0.0626	0.0510	0.0559	0.0849	0.0479	0.0671
MoNF	**0.1715**	**0.1649**	**0.0448**	**0.0391**	**0.0391**	**0.0363**	**0.0229**	**0.0220**	**0.0310**	**0.0334**
Black Scholes	0.4950	0.1840	0.3676	0.0994	0.3498	0.0889	0.4717	0.1705	0.6185	0.4270
Heston	0.4228	0.1066	0.3472	0.0661	1.8010	0.4541	1.4452	0.3736	0.6884	1.0936
Kou Jump	4.0404	0.2047	1.1308	0.0602	1.0607	0.0471	7.1047	0.1038	6.1269	0.2024
Merton Jump	1.0406	0.1824	0.3676	0.0994	0.3498	0.0889	1.4126	0.1100	0.5640	0.2514
Variance Gamma	0.3924	0.0811	0.2389	0.0469	0.1937	0.0344	0.2279	0.0323	0.5670	0.0904
MNN	1.0287	0.1906	0.3987	0.1023	0.3214	0.0934	1.3352	0.1054	0.5568	0.2459
GNN	0.4204	0.1099	0.3544	0.0633	2.1005	0.4199	1.2218	0.3088	0.7339	0.9670
MoNF (Linear)	0.5706	0.0951	0.3490	0.0626	0.1620	0.0854	0.1184	0.0385	0.6351	0.1691
MoNF	**0.1706**	**0.0289**	**0.1564**	**0.0221**	**0.0982**	**0.0222**	**0.1084**	**0.0185**	**0.3130**	**0.0564**

Table 2: Option Pricing Performance (Mean Absolute Error): Interpolation Setting (Top) and Extrapolation Setting (bottom)

of contracts for each asset and average prices can be found in Table. 1

4.1.1 Train-test split

The dataset for each individual experiment is formed by options with the same maturity on every trading day, and we have around $120,000$ datasets/experiments. On average, each dataset has 300 option contracts/samples. For those samples, we have two strategies to split them into the training and testing sets:

Interpolation We sort all options by their strike prices, and choose the even-indexed ones as training and odd-indexed ones as testing.

Extrapolation We choose 50% all options with the strike prices near the current asset price for the training, and leave the remaining options with strike prices far away from the current asset price for the testing.

The illustration of these two splits can be found in Fig. 2.

4.2 IMPLEMENTATION

We implement our method using PyTorch [Paszke et al., 2019], and tune the hyper-parameters using the data from the day before the starting date then keep them fix for the whole five years' experiments. For the main experiments, we use a mixture model with 8 NFs and each NF has 4 layers, and the reference distribution is a single 4-layer NF. We also evaluate the performance for MoNF with identify function instead of the proposed activation function, denoted as MoNF (Linear). Note that, the model is actually reduced to Gaussian Mixture Model (GMM), as the linear transformation of Gaussian is still Gaussian.

For neural network based methods, we choose Modular Neural Networks (MNN) [Gradojevic et al., 2009] and Gated Neural Networks (GNN) [Yang et al., 2017], which partially meet the constraints. For mathematical finance baselines, we choose Black–Scholes [Black and Scholes, 1973], Heston [Heston, 1993], Variance Gamma [Madan et al., 1998], Kou Jump [Kou, 2002], and Merton Jump [Matsuda, 2004], as they represent different diffusion and jump process designs.

All models are trained to minimise the mean absolute error of the real market v.s. predicted prices, and the true difference over prices is reported. This is different from some studies that train and evaluate their methods using the normalised price (e.g., option price divided by asset price), as that may not reflect the performance in trading.

4.3 RESULTS ANALYSIS

In option pricing, we care both the training and testing error, because training error reflects how well the model explains market behaviour, and the testing error reflects how well the model performs if we ask for a quote of an unseen strike

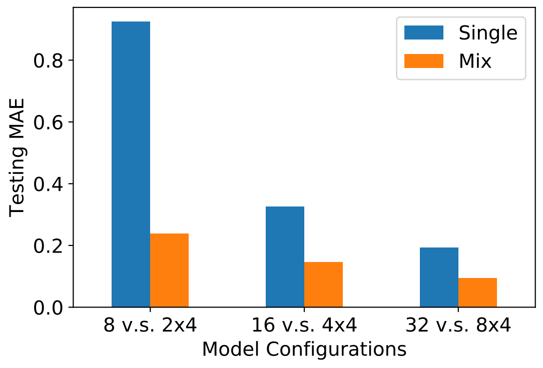


Figure 3: Single NF models vs. Mixture of NFs (Test-MAE)

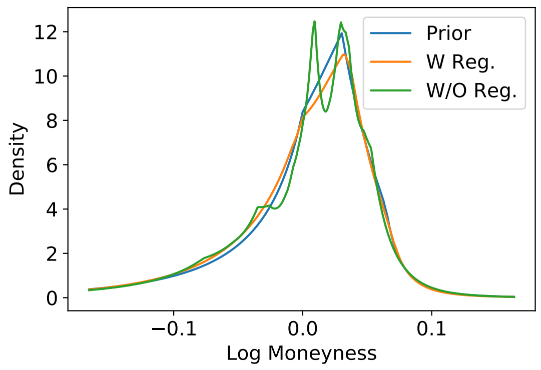


Figure 4: The effect of regularisation on MoNF.

price. In real market data, the strike prices are some common numbers (e.g., 50, 100, 150), and a practical use of option pricing is to estimate the price for an arbitrary strike (e.g., 76.54). From the summary of results in Table 2, we can easily see the advantage of the proposed method over other baselines. Interestingly, other neural network based methods, such as MNN and GNN, do not outperform the mathematical finance models constantly, which use significantly fewer parameters. We further investigate why this happened. Both MNN and GNN are designed for call option pricing, and the designs are made to meet some (not all) of the constraints so it is not easy to adapt the same design for put options, therefore, we have to convert put options to call options via put-call parity, and convert them back after pricing. However, the re-converted put options have much larger errors, even though their converted call options are well fitted by MNN and GNN. This is reasonable: put-converted call options are usually much cheaper than their original prices, so the errors over put-converted call options will be amplified when converting back to put options.

4.3.1 Is mixture model necessary?

One may question that whether we really need a mixture of normalizing flows, or the similar performance can be achieved by using a single but deeper NF model. Here we evaluate the model performance for three pairs of models: (i) an 8-layer NF v.s. a 4×2-layer MoNF (ii) a 16-layer NF v.s. a 4×4-layer MoNF (iii) a 32-layer NF v.s. a 8×4-layer MoNF. The results can be found in Fig. 3. Increasing the number of parameters helps to reduce the testing error in both cases, but it is more effective to use a mixture of NFs models rather than a single deep NF model. Besides, we have also found that the deep single model is more sensitive to initialisation empirically because of the numerical stability issues when more layers/transformations are added, so the mixture of shallower normalising flows is preferred.

4.3.2 Regularisation

We investigate the effect of the regularisation term (Sec 3.3.4). First, we train a single NF model and use it as a prior distribution. Then, we train our model with/without the regularisation term and plot the risk neutral density in Fig. 4. We can see that the density without regularisation tends to be non-smooth, due to its nature of being a mixture model. In contrast, the density with regularisation is much closer to the single NF model's. We also compare the performance of MoNF without regularisation on S&P500 index option. As we can see in Table 3, the performance is improved slightly without regularization. We reiterate that the model without regularization does not invalidate any constraints as the density is still valid despite being non-smooth. A more realistic use of regularization is to form a rather simple density function, e.g., a uni-modal distribution, so one can answer the questions like "what is the option inferred *mode* of future asset price from market data".

		Test	Train
Interpolation	MoNF (w. Reg.)	0.0310	0.0334
	MoNF (w.o. Reg)	**0.0301**	**0.0290**
Extrapolation	MoNF (w. Reg.)	**0.3130**	0.0564
	MoNF (w.o. Reg)	0.3224	**0.0519**

Table 3: Impact of regularization (S&P 500, MAE).

5 CONCLUSION

In this work, we presented a European option pricing model based on the normalizing flows. It achieved excellent performance in terms of fitting the market data as well as generalising to unseen strike prices. More crucially, it met all conditions for an economically rational model, so we are assured that it will behave rationally in out-of-distribution scenarios and rare events. Finally, we demonstrate that neural networks for risk neutral density could potentially be a better choice than directly modelling the pricing function. For the future work, we plan to extend the model from pricing curves to pricing surfaces.

Disclaimer: All authors are faculty. Neither graduate students nor small animals were hurt while producing this paper.

References

Y. S. Abu-Mostafa. Learning from hints in neural networks. *Journal of Complexity*, 6(2):192–198, 1990.

Damien Ackerer, Natasa Tagasovska, and Thibault Vatter. Deep smoothing of the implied volatility surface. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Bhupinder Bahra. Implied risk-neutral probability density functions from option prices: theory and application. Bank of england working papers, Bank of England, 1997. URL `https://EconPapers.repec.org/RePEc:boe:boeewp:66`.

Ole E. Barndorff-Nielsen. Normal inverse gaussian distributions and stochastic volatility modelling. *Scandinavian Journal of Statistics*, 24(1):1–13, 1997.

Fishcer Black and Myron Scholes. The pricing of options and corporate liabilities. *Journal of political economy*, 81 (3):637–654, 1973.

Peter W. Buchen and Michael Kelly. The maximum entropy distribution of an asset inferred from option prices. *The Journal of Financial and Quantitative Analysis*, 31(1): 143–159, 1996.

Peter Carr and Helyette Geman. The fine structure of asset returns: An empirical investigation. *The Journal of Business*, 75(2):305–332, 2002.

Peter Carr, Hélyette Geman, Dilip B. Madan, and Marc Yor. Stochastic volatility for lévy processes. *Mathematical Finance*, 13(3):345–382, 2003.

Laurent Dinh, David Krueger, and Yoshua Bengio. Nice: Non-linear independent components estimation. In *International Conference on Learning Representations (ICLR) Workshop*, 2015.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using real nvp. In *International Conference on Learning Representations (ICLR)*, 2017.

Charles Dugas, Yoshua Bengio, Francois Belisle, Claude Nadeau, and Rene Garcia. Incorporating second-order functional knowledge for better option pricing. In *Neural Information Processing Systems (NIPS)*, 2000.

Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

René Garcia and Ramazan Gençay. Pricing and hedging derivative securities with neural networks and a homogeneity hint. *Journal of Econometrics*, 94(1):93–115, 2000.

N. Gradojevic, R. Gencay, and D. Kukolj. Option pricing with modular neural networks. *IEEE Transactions on Neural Networks*, 20(4):626–637, 2009.

Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. *International Conference on Learning Representations (ICLR)*, 2019.

Alex Graves. Stochastic backpropagation through mixture density distributions. *CoRR*, abs/1607.05690, 2016.

Steven L. Heston. A closed-form solution for options with stochastic volatility with applications to bond and currency options. *Review of Financial Studies*, 6:327–343, 1993.

Emiel Hoogeboom, Jorn Peters, Rianne van den Berg, and Max Welling. Integer discrete flows and lossless compression. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

James M. Hutchinson, Andrww W. Lo, and Tomaso Poggio. A nonparametric approach to pricing and hedging derivative securities via learning networks. *The Journal of Finance*, 49(3):851–889, 1994.

M. Jeanblanc, M. Yor, and M. Chesney. *Mathematical Methods for Financial Markets*. Springer Finance. Springer-Verlag London Ltd., 2009.

Durk P Kingma and Prafulla Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Neural Information Processing Systems (NeurIPS)*, 2018.

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Neural Information Processing Systems (NIPS)*, 2016.

S. G. Kou. A jump-diffusion model for option pricing. *Management Science*, 48(8):1086–1101, 2002.

Gabriel Loaiza-Ganem, Yuanjun Gao, and John P. Cunningham. Maximum entropy flow networks. In *International Conference on Learning Representations (ICLR)*, 2017.

Dilip B. Madan, Peter Carr, and Eric C. Chang. The variance gamma process and option pricing. *European Finance Review*, 2:79–105, 1998.

Mary Malliaris and Linda Salchenberger. A neural network model for estimating option prices. *Applied Intelligence*, 3(3):193–206, 1993.

Kazuhisa Matsuda. Introduction to merton jump diffusion model, 2004.

Robert C. Merton. Option pricing when underlying stock returns are discontinuous. *Journal of Financial Economics*, 3:125–144, 1976.

Ana Margarida Monteiro, Reha H. Tutuncu, and Luis N. Vicente. Recovering risk-neutral probability density functions from options prices using cubic splines and ensuring nonnegativity. *European Journal of Operational Research*, 187(2):525–542, 2008.

Cassio Neri and Lorenz Schneider. Maximum entropy distributions inferred from option portfolios on an asset. *Finance and Stochastics*, 16(2):293–318, 2012.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *Neural Information Processing Systems (NIPS)*, 2017.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *CoRR*, abs/1912.02762, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, 2019.

Johannes Ruf and Weiguan Wang. Neural networks for option pricing and hedging: a literature review. *Journal of Computational Finance*, 24(1):1–46, 2020.

Yongxin Yang, Yu Zheng, and Timothy M. Hospedales. Gated neural networks for option pricing: Rationality by design. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2017.