Forecasting and Pricing with Volatility Integrated Recurrent Neural Networks

Yaw Asante

July 2025

1 Introduction

This proposal outlines a comprehensive framework to construct a neural network that embeds volatility dynamics for forecasting and option valuation. The benchmark will be the Affine GARCH(1,1) model of Heston and Nandi (2000), which has closed-form expressions for option prices under a risk-neutral Esscher transform, and the component Heston and Nandi model of Christoffersen et. al (2010). Let $t \in [0, T]$, our goal is to compare the forecasting and pricing abilities classical HN-based models with the volatility integrated neural networks.

2 Benchmark Models

2.1 Heston and Nandi GARCH(1,1)

We use the return and variance dynamics from Heston and Nandi (2000):

$$R_{t+1} \equiv \ln\left(\frac{S_{t+1}}{S_t}\right) = r - d + \lambda h_{t+1} + \sqrt{h_{t+1}} z_{t+1} \tag{1}$$

$$h_{t+1} = \omega + \beta h_t + \alpha\left(z_t - \gamma\sqrt{h_t}\right)^2 \tag{2}$$

The unconditional variance is:

$$\bar{\omega} = \frac{\omega + \alpha}{1 - \alpha\gamma^2 - \beta}$$

From this, we can express h_{t+1} recursively in terms of σ^2:

$$h_{t+1} = \bar{\omega} + \phi(h_t - \bar{\omega}) + a\left(z_t^2 - 1 - 2\gamma\sqrt{h_t} z_t\right)$$

So the reparmeterized equation becomes:

$$R_{t+1} = r - d + \lambda h_{t+1} + \sqrt{h_{t+1}} z_{t+1}, \quad z_t \sim \mathcal{N}(0, 1) \tag{3}$$

$$h_{t+1} = \bar{\omega} + \phi(h_t - \bar{\omega}) + \alpha\left(z_t^2 - 1 - 2\gamma\sqrt{h_t} z_t\right) \tag{4}$$

With persistence defined as $\phi = \beta + \alpha\gamma^2$, the model supports analytical derivation of forward variance and its variance (Eqs. (5)–(6)):

$$\mathbb{E}_t[h_{t+k}] = \bar{\omega} + \phi^{k-1}(h_{t+1} - \bar{\omega})$$
$$\mathrm{Var}_t[h_{t+2}] = 2\alpha^2 + 4\alpha^2\gamma^2 h_{t+1}$$

It also allows closed-form expressions for conditional return-variance dependence (Eq. (7)–(8)):

$$\mathrm{Cov}_t[R_{t+1}, h_{t+2}] = -2\alpha\gamma h_{t+1}, \quad \mathrm{Corr}_t[R_{t+1}, h_{t+2}] = \frac{-2\gamma\sqrt{h_t}}{\sqrt{2 + 4\gamma^2 h_t}}$$

2.2 Component Heston and Nandi GARCH(1,1)

Christoffersen et al(2010) models the long run variance of (4) with another autoregressive model, to obtain the short run component and long run component like so:

$$R_{t+1} = r - d + \lambda h_{t+1} + \sqrt{h_{t+1}} z_{t+1}, \quad z_t \sim \mathcal{N}(0,1) \tag{5}$$
$$h_{t+1} = q_{t+1} + \phi(h_t - q_t) + \alpha\left(z_t^2 - 1 - 2\gamma_1\sqrt{h_t} z_t\right) \tag{6}$$
$$q_{t+1} = \bar{\omega} + \rho(q_t - \bar{\omega}) + \varphi\left(z_t^2 - 1 - 2\gamma_2\sqrt{h_t} z_t\right) \tag{7}$$

This model collapses to the Heston and Nandi model above when $\rho = \varphi = 0$. We seek to enhance volatility by integrating (4) and (6) into the GRU and LSTM recurrent models. This way, volatility directly controls information gates. We start with a single gate integration to see if this concept works, then we can extend it to multi gates, multi-volatility components, etc.

3 Recurrent Neural Networks

Recurrent neural networks are a neutral network architecture that can be viewed as a feed-forward network with feedback connections. It is used to model temporal data. The basic recurrent network is a shallow network that loses information easily and suffer for the gradient vanishing problem. There are newer, more sophisticated recurrent networks that are design to mitigate these issues.

3.1 Long Short-Term Memory (LSTM) Networks

The LSTM incorporates specialized gates to help in deciding which information to keep and which ones to discard (forget), thus enabling the network to possess "memory". The lstm

can be written as:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}\eta_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}\eta_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}\eta_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}\eta_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
\eta_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

where $\odot$ is the Hadamard product. It explicitly has a separate memory cell c_t in addition to the hidden state η_t. The model uses three gates: The input gate i_t, the forget gate f_t and output gate o_t to regulate what information is written to, erased from, and read from the memory cell. This explicit memory pathway allows LSTMs to preserve long-term dependencies with fine-grained control.

3.2 Gated Recurrent Unit (GRU)

This is an abridge version of the LSTM in that it does not have a separate cell state. It merges memory and hidden state into a single vector η_t. This make is computationally efficient and easier to train because it has fewer parameters. It also performs comparably to the LSTM in many applications.

$$
\begin{aligned}
r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}\eta_{t-1} + b_{hr}) \\
u_t &= \sigma(W_{iu}x_t + b_{iu} + W_{hu}\eta_{t-1} + b_{hu}) \\
n_t &= \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}\eta_{t-1} + b_{hn})) \\
\eta_t &= (1 - u_t) \odot n_t + u_t \odot \eta_{t-1}
\end{aligned}
$$

4 Volatility Integration

We seek a hybrid model where the volatility dynamics in integrated into the LSTM and GRU models defined above. Their biases are currently constants, we seek to make them dynamic, with the dynamics governed by the volatility equations (4) and (6). We start out by only making one of the biases dynamic to test out this concept and see if it works well. For the GRU, inject the volatilty into the update gate (u_t). For LSTM, we inject the volatility into the forget gate f_t. This way, our LSTM and GRU models respectively becomes:

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}\eta_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + W_{hf}\eta_{t-1} + b_t) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}\eta_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}\eta_{t-1} + b_{ho}) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
\eta_t &= o_t \odot \tanh(c_t)
\end{aligned}
$$

Here $b_t = \gamma_f h_t$, and h_t is from (4) and (6).

For the GRU, we have

$$\begin{aligned} r_t &= \sigma(W_{ir}x_t + b_{ir} + W_{hr}\eta_{t-1} + b_{hr}) \\ u_t &= \sigma(W_{iu}x_t + W_{hu}\eta_{t-1} + b_t) \\ n_t &= \tanh(W_{in}x_t + b_{in} + r_t \odot (W_{hn}\eta_{t-1} + b_{hn})) \\ \eta_t &= (1 - u_t) \odot n_t + u_t \odot \eta_{t-1} \end{aligned}$$

where $b_t = \gamma_u h_t$.

We want to train by MLE. The likelihood for the Heston-Nandi and Component Heston-Nandi are the same since they both share the same returns equation. This is:

$$\log \mathcal{L} = \sum_{t=1}^{T} \left[-\frac{1}{2}\log(2\pi) - \frac{1}{2}\log h_t - \frac{(R_t - r + d - \lambda h_t)^2}{2h_t} \right] \tag{8}$$

The parameters space of the HN model is $\Theta = [\bar{\omega}, \alpha, \phi, \lambda, \gamma]$ and $\Theta = [\bar{\omega}, \alpha, \phi, \lambda, \gamma_1, \gamma_2, \varphi, \rho]$. For the option pricing, we rely on Monte Carlo Simulations, with 10,000 paths and 250 maturity.

5 NN based HN and CHN

After integrating the HN and CHN into the GRU and LSTM, we hope to enhance the returns equations. How NN based Heston-Nandi becomes.

$$R_{t+1} = r - d + \lambda h_{t+1} + \sqrt{\eta_{t+1}} z_{t+1}, \quad z_t \sim \mathcal{N}(0,1) \tag{9}$$

$$h_{t+1} = \bar{\omega} + \phi(h_t - \bar{\omega}) + \alpha\left(z_t^2 - 1 - 2\gamma\sqrt{h_t} z_t\right) \tag{10}$$

$$\eta_{t+1} = RNN(X_t), \tag{11}$$

where RNN stands for either GRU or LSTM and X_t is the input vector. The corresponding component model becomes

$$R_{t+1} = r - d + \lambda h_{t+1} + \sqrt{\eta_{t+1}} z_{t+1}, \quad z_t \sim \mathcal{N}(0,1) \tag{12}$$

$$h_{t+1} = q_{t+1} + \phi(h_t - q_t) + \alpha\left(z_t^2 - 1 - 2\gamma_1\sqrt{h_t} z_t\right) \tag{13}$$

$$q_{t+1} = \bar{\omega} + \rho(q_t - \bar{\omega}) + \varphi\left(z_t^2 - 1 - 2\gamma_2\sqrt{h_t} z_t\right) \tag{14}$$

$$\eta_{t+1} = RNN(X_t), \tag{15}$$

5.1 Input Considerations

It is very important to consider what will be the input choice here. Since the setup is going to be used for option pricing, every input (observed or engineered) should be calculable for simulation. If an input can only be obtained from observed data, then it will be interesting to see how the monte carlo simulation is done with the inclusion of such an input. We do not place restrictions on the inputs used, only that it should be viable (or tractable) for monte carlo simulation too.

6 Extensions

If this Phase 1 works well, then we can include the volatility equation in all gets of the RNNs. We can also include multiple volatility sources, models, data, etc.