

## Outline

A Neural Network Based Framework for Financial Models Summary .....	2
4.3 Backward Pass (Heston Model) .....	2
How does this relate to our research? .....	2
Sampling Training Data .....	2
Averaged performance of the Backward pass of the CaNN: .....	2
Summary of Joint Calibration Artificial Neural Networks .....	3
Goals & Designs .....	3
Risk Neutralization for One-Component Gaussian Models .....	3
NGARCH(1, 1) .....	3
Duan .....	4
HN-GARCH(1, 1) .....	4
Input Data .....	5
Joint Calibration Neural Network .....	5
Essential Equations .....	7
Return Log Likelihood: .....	7
Options Log Likelihood: .....	7
Joint Log Likelihood .....	7
Training Data Generation .....	8

# A Neural Network Based Framework for Financial Models

## Summary

### 4.3 Backward Pass (Heston Model)

$\sigma_{\text{imp}} \mid K, \tau, S_0, r \rightarrow \text{Heston-CaNN} \rightarrow \rho, k, v_0, \bar{v}, \gamma$

#### How does this relate to our research?

- We hope to be able to use historical asset prices and option prices to be able to calibrate our parameters. This mean, our input depends on the different parameters that make these prices/returns, which would be strike prices, initial price, rate, time to maturity and the implied volatility.
- We then send these input to the CaNN where it would train the model
- After we train the model, we can use our Joint objective function to return the calibrated parameters, where in our case, since we are just caring about the GARCH(1, 1) model would be  $\omega, \alpha, \beta$ .

#### Sampling Training Data

Found in Table 5 of the paper:

	Parameters	Range	Samples
Market data	Moneyness, $m = \frac{S_0}{K}$	[0.85, 1.15]	5
	Time to maturity, $\tau$	[0.5, 2.0](year)	7
	Risk free rate, $r$	0.03	Fixed,
	European call/put price, $\frac{V}{K}$	(0.0, 0.6)	•
Black-Scholes	Implied Volatility	(0.2, 0.5)	35

We can use this as a way to sample all our different input parameters, and be able to measure how more accurate the model becomes and its calibration.

During the calibration, they use the total squared error measure  $J(\Theta)$ :

$$J(\Theta) = \sum \omega (\sigma_{\text{imp}}^{\text{ANN}} - \sigma_{\text{imp}}^*)^2 + \bar{\lambda} |\Theta|$$

#### Averaged performance of the Backward pass of the CaNN:

- Need to list CPU and GPU spec
  - OS: Linux pop-os 6.6.10-76060610-generic
  - CPU: AMD Ryzen 5 5600G
  - GPU: Radeon 6600

Abosolute deviation from  $\theta^*$ , Error measure and computational cost.

Error Measure:  $J(\Theta)$ , MJ and Data Points

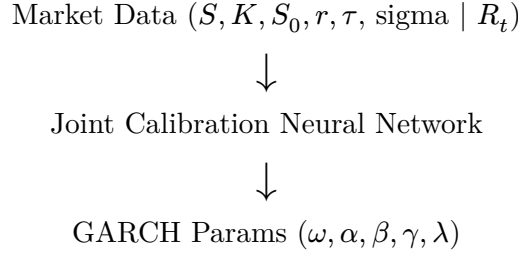
Computational Cost: CPU, GPU time and Function evaluations

# Summary of Joint Calibration Artificial Neural Networks

## Goals & Designs

Our goal is to be able to use two forms of input data to be able to calibrate an optimum parameters of the GARCH(1, 1) model. This will be calibrated using joint calibration which will take into account the log likelihood of returns and option prices to be able to consider both physical and risk neutral measures.

The design will make use of a backward pass artificial neural network, where we will do the following:



Where we consider the parameters under **P** measure:

- $S$ : Price (a time series of asset prices eg. (10-year daily prices))
- $K$ : Strike Price
- $S_0$ : Initial Price
- $r$ : Risk-free rate (fixed)
- $\sigma$ : Implied Volatility
- $\tau$ : Time to maturity

And we consider the following physical measure:

- $R_t$ : Log return at time  $t$  (several GARCH models)

Our goal is to be able to use the Joint Calibration to have the most optimum calibrated GARCH parameters that takes into account both measures. This will come into the calibration phase, where the idea is to utilize the Joint Calibration formula as the objective function for minimization.

## Risk Neutralization for One-Component Gaussian Models

- $Q$ : Risk-Neutral Measure
- $P$ : Physical Measure

Using the Radon-Nikodym derivative, we can convert the physical measure to the risk-neutral measure. Let  $z_t$  i.i.d  $N(0, 1)$ , then  $\gamma_t = \frac{1}{2}h_t$  since  $\exp(\gamma_t) = E_{t-1}[\exp(\varepsilon_t)]$

The Radon-Nikodym derivative is defined as:

$$\frac{dQ}{dP} \mid F_t = \exp\left(-\sum_{i=1}^t \left(\frac{\mu_i - r_i}{h_i} \varepsilon_i + \frac{1}{2} \left(\frac{\mu_i - r_i}{h_i}\right)^2 h_i\right)\right)$$

## NGARCH(1, 1)

For NGARCH(1, 1) using  $\varepsilon_t^* = \varepsilon_t + \mu_t - r_t$ , the volatility process under  $Q$  becomes:

$$\begin{aligned}
h_t &= \omega + \beta h_{t-1} + \alpha(\varepsilon_{t-1}^* - \mu_{t-1} + r_{t-1})^2 \Rightarrow \varepsilon_t^* \mid F_t \sim N(0, h_t) \\
R_t &\equiv \ln\left(\frac{S_t}{S_{t-1}}\right) = r_t - \frac{1}{2}h_t + \varepsilon_t^* \Rightarrow \varepsilon_t^* \mid F_{t-1} \sim N(0, h_t) \\
E^Q\left[\frac{S_t}{S_{t-1}} \mid F_{t-1}\right] &= \exp(r_t)
\end{aligned}$$

## Duan

The Physical GARCH Model Duan (1995) comes in the following form:

$$\begin{aligned}
R_t &\equiv \ln\left(\frac{S_t}{S_{t-1}}\right) = r_t + \lambda\sqrt{h_t} - \frac{1}{2}h_t + \varepsilon_t \\
h_t &= \omega + \beta h_{t-1} + \alpha \varepsilon_{t-1}^2
\end{aligned}$$

Assume:

- $\lambda$ : price of risk (const.)
- $\mu_t = r_t + \lambda\sqrt{h_t}$  or  $\lambda = \frac{\mu_t - r_t}{\sqrt{h_t}}$

This corresponds to the following RN-Derivative:

$$\frac{dQ}{dP} \mid F_t = \exp\left(-\sum_{i=1}^t \left(\frac{\varepsilon_i}{\sqrt{h_i}}\lambda + \frac{1}{2}\lambda^2\right)\right)$$

With risk-neutral innovations:

$$\begin{aligned}
\varepsilon_t^* &= \varepsilon_t + \mu_t - r_t \\
&= \varepsilon_t + \lambda\sqrt{h_t}
\end{aligned}$$

The Risk-Neutral GARCH becomes:

$$\begin{aligned}
R_t &\equiv \ln\left(\frac{S_t}{S_{t-1}}\right) = r - \frac{1}{2}h_t + \varepsilon_t^* \\
h_t &= \omega + \beta h_{t-1} + \alpha(\varepsilon_{t-1}^* - \lambda\sqrt{h_{t-1}})^2
\end{aligned}$$

## HN-GARCH(1, 1)

Starting with the following model of Heston and Nandi (2000):

$$\begin{aligned}
R_t &\equiv \ln\left(\frac{S_t}{S_{t-1}}\right) = r + \lambda h_t + \varepsilon_t \\
h_t &= \omega + \beta h_{t-1} + \alpha(z_{t-1} - c\sqrt{h_{t-1}})^2
\end{aligned}$$

Assume  $r_t = r, \mu_t = r + \lambda h_t + 0.5h_t$

RN-Derivative:

$$\frac{dQ}{dP} \mid F_t = \exp \left( - \sum_{i=1}^t \left( \left( \lambda + \frac{1}{2} \right) \varepsilon_i + \frac{1}{2} \left( \lambda + \frac{1}{2} \right)^2 h_i \right) \right)$$

$$\varepsilon_t^* = \varepsilon_t + \lambda h_t + 0.5 h_t$$

$$R_t \equiv \ln \left( \frac{S_t}{S_{t-1}} \right) = r - \frac{1}{2} h_t + \varepsilon_t^*$$

$$h_t = \omega + \beta h_{t-1} + \alpha \left( z_{t-1}^* - \left( c + \lambda + \frac{1}{2} \right) \sqrt{h_{t-1}} \right)^2$$

## Input Data

So firstly we need to discuss how we get our input parameters?

Risk-Neutral Measure:

- American Option Prices

Physical Measure:

- Historical Asset Prices (change in difference to get log return)
  - Where:  $R_t \equiv \ln \left( \frac{S_t}{S_{t-1}} \right) = \mu_t - \gamma_t + \varepsilon_t$
  - $\mu_t$ : Conditional mean of the returns at time  $t$
  - Assume  $\gamma_t$  is defined from  $\exp(\gamma_t) = E_{t-1}[\exp(\varepsilon_t)]$
  - $\varepsilon_t$ : the normal innovation at time  $t$  where  $\varepsilon_t \mid F_{t-1} \sim N(0, h_t)$
  - $h_t$ : Conditional variance at time  $t$ 
    - $h_t = \omega + \alpha \varepsilon_{t-1}^2 + \beta h_{t-1}$

## Joint Calibration Neural Network

This neural network will be written in Python with a minimum required version of 3.10, and will require the following dependencies:

[[tool.poetry.dependencies](#)]

python = "^3.10"

numpy = "^1.26.4"

pandas = "^2.2.2"

arch = "^7.0.0"

scikit-learn = "^1.5.0"

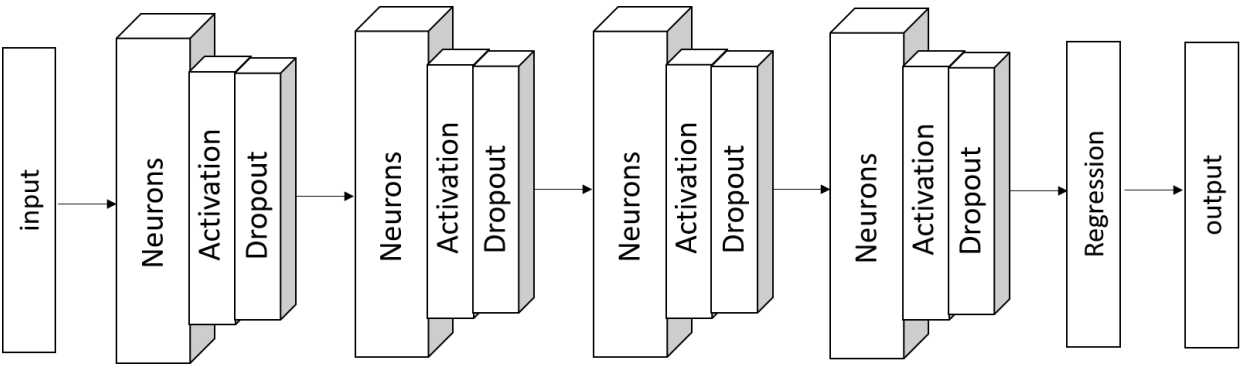
matplotlib = "^3.9.0"

scipy = "^1.13.1"

The architecture currently will follow similarly to the paper, as the following:

Parameters	Options
Hidden Layers	4
Neurons(each layer)	200
Activation	ReLU
Dropout rate	0.0
Batch-normalization	No
Initialization	Glorot_uniform

Parameters	Options
Optimizer Batch size	Adam 1024



Where we then hope in the regression phase, to be able to use the Joint Calibration formula in a batching method to be able to calibrate our parameters for the GARCH model.

## Essential Equations

These are from GARCH Option Valuation Theory and Evidence:

### Return Log Likelihood:

$$\ln L^R \propto -\frac{1}{2} \sum_{t=1}^T \left\{ \ln(h(t)) + \frac{(R(t) - \mu_t - \gamma_t)^2}{h(t)} \right\}$$

where:

- $h(t)$  is the conditional variance
- $R(t)$  represents the return at time  $t$
- $\mu_t$  is the conditional mean of the returns at time  $t$
- $\gamma_t$  is an adjustment term

### Options Log Likelihood:

$$\ln L^O \propto -\frac{1}{2} \sum_{i=1}^N \left\{ \ln(\text{IVRMSE}^2) + \left( \frac{e_{i,t}}{\text{IVRMSE}} \right)^2 \right\}$$

Implied Volatility root mean squared error (IVRMSE) loss function follows:

$$\text{IVRMSE} \approx \sqrt{\frac{1}{N_T} \sum_{i,t} e_{i,t}^2}$$

where:

- $N_T$ : total number of option prices in the sample

The verga weighted option error follows as:

$$e_{i,t} = \frac{C_{i,t} - C_{i,t}(h_t(\xi^*))}{\text{Vega}_{i,t}}$$

We may also choose to use relative error if Vega is hard or expensive to calculate

where:

- $\text{Vega}_{i,t}$  is the Black-Scholes sensitivity of the option prices with respect to volatility
- $\xi^*$  is the vector of risk-neutral parameters to be estimated
- $C_{i,t} - C_{i,t}(h_t(\xi^*))$ : The corresponding implied volatility from the option price.

### Joint Log Likelihood

$$L_{\text{joint}} = \frac{T + N_T}{2} \frac{L^R}{T} + \frac{T + N_T}{2} \frac{L^O}{N}$$

where:

- $T$  is the number of days in the return sample
- $N_T$  is the total number of option contracts

## Training Data Generation

### Current Idea

1. Given a set of parameters for GARCH (in Physical measure)
2. Given the initial asset price  $S_0$ , use Monte Carlo method to simulate a path of asset prices,  $S_1, S_2, \dots, S_N$ , with say  $N = 500$  (Under  $\mathbf{P}$  measure)
3. Select last 30-50 days on the path, for each day, use the selected asset price (under  $\mathbf{Q}$ ) as the initial price to generate American option prices with various strike prices (11-17) and maturities (7 days to 1 year). **Pay attention to the transformation from the physical measure to the risk-neutral measure.**

### Pseudo Code

```
# Pseudo Code of Pricing American Options By the Willow Tree and Monte Carlo Method
# 1. Initialize option parameters
T = ... # Time to maturity
K = ... # Strike price
r = ... # Risk-free rate
S_0 = ... # Initial asset price
N = ... # Number of paths
delta = T/N # Time step
h0 = ... # Initial volatility
# 2. Initialize HN-GARCH parameters under P Measure
alpha = ...
beta = ...
omega = ...
gamma = ...
lambda = ...
# 3. Initialize Willow Tree parameters
m_h = ...
m_ht = ...
m_x = ...
gamma_h = ...
gamma_x = ...
# 4. Construct the willow tree for ht
hd, qhd = genhDelta(h0, beta, alpha, gamma, omega, m_h, gamma_h)
nodes_ht = TreeNodes_ht_HN(m_ht, hd, qhd, gamma_h, alpha, beta, gamma, omega, N + 1)
P_ht_N, P_ht = Prob_ht(nodes_ht, h0, alpha, beta, gamma, omega)
# 5. Construct the willow tree for Xt
nodes_Xt, mu, var, k3, k4 = TreeNodes_logSt_HN(m_x, gamma_x, r, hd, qhd, S0, alpha, beta, gamma, omega, N)
q_Xt, P_Xt, tmpHt = Prob_Xt(nodes_ht, qhd, nodes_Xt, S0, r, alpha, beta, gamma, omega)
nodes_S = np.exp(nodes_Xt)
# 6. Generate Data (can possibly be done in parallel or concurrently)
days = ...
A_prices = np.zeros(days) # American option prices
A_sig = np.zeros(days) # Implied volatility
A0_prices = np.zeros(days) # Price of option using model parameters

for i in range(days): # Days on the path
    CorP = -1 # Call or Put
    A_sig[i], A_prices[i], A0_prices[i] = impVol_HN(r, lambda, omega, beta, alpha,
                                                    ...gamma, h0, S0, K, T, N, m_h, m_x, cor_p)
```



Functions to be aware of:

- **genhDelta**: Generates the discrete values and probabilities of a std normal distribution that are used to construct a Willow tree for the conditional variance in the HN model.
- **TreeNodes\_ht\_HN**: Constructs the Willow tree for the conditional variance in the HN model.
- **Prob\_ht**: Calculates the transition probabilities of the nodes in the Willow tree for the conditional variance in the HN model.
- **TreeNodes\_logSt\_HN**: Constructs the nodes of the Willow tree for the log asset price in the HN model, as well as the first four moments.
- **Prob\_Xt**: Calculates the transition probabilities of the nodes in the Willow tree for the log asset price in the HN model.
- **impVol\_HN**: Calculates the American option price, the implied volatility, and the option price using the model parameters in the HN model.

### Link to Code

Once Python code is able to produce similar results to Matlab, it will have an initial Github release, with this link. It will be ZIP file under the name JC\_WT\_DataGen

### Alpha 1 Release