Outline

A Neural Network Based Framework for Financial Models Summary

4.3 Backward Pass (Heston Model)

σ_{imp} | K, τ, S_0, r -> Heston-CaNN -> ρ, k, v_0, $\overline{v}$, γ

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 ω, α, β.

Sampling Training Data

Found in Table 5 of the paper:

	Parameters	**Range**	**Samples**
Market data	Moneyness, $m = \frac{S_0}{K}$ Time to maturity, τ Risk free rate, r European call/put price, $\frac{V}{K}$	$[0.85, 1.15]$ $[0.5, 2.0]$(year) 0.03 $(0.0, 0.6)$	5 7 Fixed, •
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 $J(\Theta)$:

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

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

Abosulute deviation from θ^*, Error measure and computational cost.
Error Measure: $J(\Theta)$, MJ and Data Points
Computational Cost: CPU, GPU time and Function evaluations

Summary of Joint Calibrarion 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:

$$\text{Market Data } (S, K, S_0, r, \tau, \text{ sigma} \mid R_t)$$
$$\downarrow$$
$$\text{Joint Calibration Neural Network}$$
$$\downarrow$$
$$\text{GARCH Params } (\omega, \alpha, \beta, \gamma, \lambda)$$

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)
- σ: Implied Volatility
- τ: 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 $\exp(\gamma_t) = E_{\text{t-1}}[\exp(\varepsilon_t)]$
The Radon-Nikodym derivative is defined as:

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

NGARCH(1, 1)

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

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

Duan

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

$$R_t \equiv \ln\left(\frac{S_t}{S_{\text{t-1}}}\right) = r_t + \lambda\sqrt{h_t} - \frac{1}{2}h_t + \varepsilon_t$$
$$h_t = \omega + \beta h_{\text{t-1}} + \alpha\varepsilon^2_{\text{t-1}}$$

Assume:
- λ: 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:

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

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:

$$R_t \equiv \ln\left(\frac{S_t}{S_{\text{t-1}}}\right) = r - \frac{1}{2}h_t + \varepsilon^*_t$$
$$h_t = \omega + \beta h_{\text{t-1}} + \alpha\left(\varepsilon^*_{\text{t-1}} - \lambda\sqrt{h_{\text{t-1}}}\right)^2$$

HN-GARCH(1, 1)

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

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

Assume $r_t = r, \mu_t = r + \lambda h_t + 0.5 h_t$
RN-Derivative:

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

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

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

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

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: $R_t \equiv \ln\left(\frac{S_t}{S_{\mathrm{t\text{-}1}}}\right) = \mu_t - \gamma_t + \varepsilon_t$
  - μ_t: Conditional mean of the returns at time t
  - Assume γ_t is defined from $\exp(\gamma_t) = E_{\mathrm{t\text{-}1}}[\exp(\varepsilon_t)]$
  - ε_t: the normal innovation at time t where $\varepsilon_t | F_{\mathrm{t\text{-}1}} \sim N(0, h_t)$
  - h_t: Conditional variance at time t
    - $h_t = \omega + \alpha\varepsilon_{\mathrm{t\text{-}1}}^2 + \beta h_{\mathrm{t\text{-}1}}$

Joint Calibration Neural Network

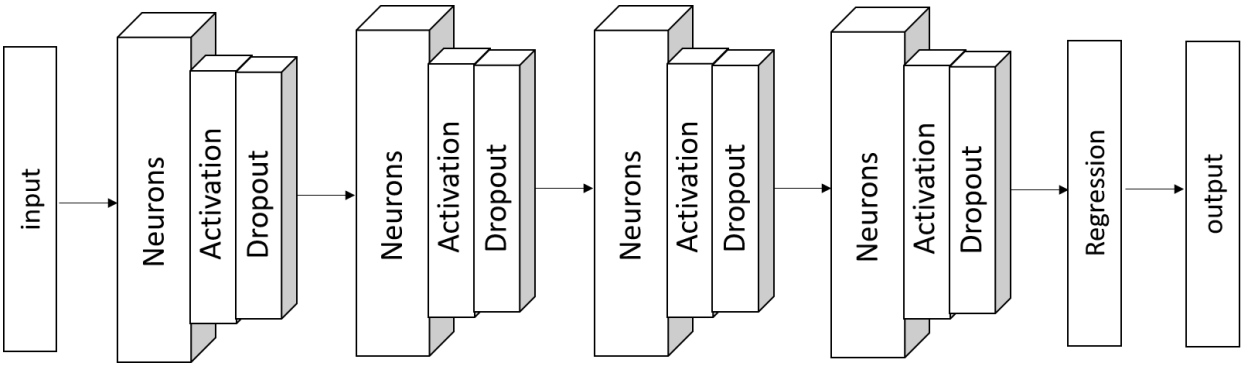
This neural network will be written in Python with a minimum required version of 3.10, and will require the following dependencies:

```
[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 simarly to the paper, as the following:

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

Parameters	Options
Optimizer	Adam
Batch size	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:

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

where:
- $h(t)$ is the conditional variance
- $R(t)$ represents the return at time t
- μ_t is the conditional mean of the returns at time t
- γ_t is an adjustment term

Options Log Likelihood:

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

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

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

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

The verga weighted option error follows as:

$$\underbrace{e_{\mathrm{i,t}} = \frac{C_{\mathrm{i,t}} - C_{\mathrm{i,t}}(h_t(\xi^*))}{\mathrm{Vega}_{\mathrm{i,t}}}}_{\text{We may also choose to use relative error if Vega is hard or expensive to calculate}}$$

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

Joint Log Likelihood

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

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, $S_1, S_2, ...S_N$, with say $N = 500$ (Under **P** measure)
3. Select last 30-50 days on the path, for each day, use the selected asset price (under **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
S0 = ...; % Initial asset price
N = ...; % Number of time steps for simulation (ex. 500)
M = ...; % Number of Monte Carlo paths
delta = T/N; % Time step
h0 = ...; % Initial volatility

% 2. Initialize HN-GARCH parameters under P Measure
alpha = ...;
beta = ...;
omega = ...;
gamma = ...;
lambda = ...;

% 3. Simulate paths using Monte Carlo simulation under P measure
% Number of paths will be N so we simulate a path of asset prices
% from S1, S2, ... S_N
points = N + 1; % number of points

Z = randn(points + 1, N);
Z1 = randn(points, N);
ht = nan(points + 1, N);
ht(1,:) = h0*ones(1,N);
Xt(1,:) = log(S0)*ones(1,N);
for i=2:points
    ht(i,:) = omega+alpha*(Z(i-1,:)-gamma*sqrt(ht(i-1,:))).^2+beta*ht(i-1,:);
    Xt(i,:) = Xt(i-1,:)+(r-0.5*ht(i,:))+sqrt(ht(i,:)).*Z(i,:);
end
ht(i+1,:) = omega+alpha*(Z(i,:)-gamma*sqrt(ht(i,:))).^2+beta*ht(i,:);
S = exp(Xt);

% 4. Risk-neutralize GARCH parameters (Q measure)
eta = ...;
omega_Q = omega / (1-2*alpha*eta);
gamma_Q = gamma*(1-2*alpha*eta);
alpha_Q = alpha /  (1-2*alpha*eta)^2;
lambda_Q = lambda*(1-2*alpha*eta);
```

```
rho_Q = lambda_Q + gamma_Q + 1/2;

% 5. Initialize Willow Tree parameters
m_h = ...;
m_ht = ...;
m_x = ...;
gamma_h = ...;
gamma_x = ...;

% 6. Construct the willow tree for ht (using Q measure parameters)
[hd, qhd] = genhDelta(h0, beta_Q, alpha_Q, gamma_Q, omega_Q, m_h, gamma_h);
nodes_ht = TreeNodes_ht_HN(m_ht, hd, qhd, gamma_h, alpha_Q, beta_Q, gamma_Q, omega_Q, N
+ 1);
[P_ht_N, P_ht] = Prob_ht(nodes_ht, h0, alpha_Q, beta_Q, gamma_Q, omega_Q);

% 7. Construct the willow tree for Xt (using Q measure parameters)
[nodes_Xt, mu, var, k3, k4] = TreeNodes_logSt_HN(m_x, gamma_x, r, hd, qhd, S_0, alpha_Q,
beta_Q, gamma_Q, omega_Q, N);
[q_Xt, P_Xt, tmpHt] = Prob_Xt(nodes_ht, qhd, nodes_Xt, S_0, r, alpha_Q, beta_Q, gamma_Q,
omega_Q);
nodes_S = exp(nodes_Xt);

% 8. Generate Data for last number of days
days_to_price = ...; % could be 50 for the last 50 days
moneyness = [0.8, 0.9, 0.95, 1, 1.05, 1.1, 1.2]; % Example: from 80% to 120% of current
price
maturities = [7/365, 1/12, 1/4, 1/2, 1]; % 7 days, 1 month, 3 months, 6 months, 1 year

A_prices = zeros(days_to_price, length(moneyness), length(maturities));
A_sig = zeros(days_to_price, length(moneyness), length(maturities));
A0_prices = zeros(days_to_price, length(moneyness), length(maturities));

for i = 1:days_to_price
    % Use the last simulated path as an example
    S_t = simulated_paths{1}(end-days_to_price+i);
    % Calculate strike prices based on current stock price
    strike_prices = S_t * moneyness;
    for j = 1:length(moneyness)
        for k = 1:length(maturities)
            CorP = -1; % Call or Put
            [A_sig(i,j,k), A_prices(i,j,k), A0_prices(i,j,k)] = impVol_HN(r, lambda_Q,
                                ... omega_Q, beta_Q, alpha_Q, gamma_Q, h0, S_t,
                                ... strike_prices(j), maturities(k), N, m_h, m_x, CorP);
        end
    end
end
```

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.

f_hhh Situation The current dependency on `f_hhh` relies on a MEX file which is compiled from C code, currently the best tool to handle this is `mkoctfile` from Octave. Currently a Matlab native code is giving the following results (from the original demo code Prices)

Windows f_hhh.mexw64	Linux f_hhh.m	Linux f_hhh.mexa64	Rel Error of f_hhh.m to Mex
2.72207972210372	25.9570397649806	2.68249096732144	853.57%
10.0008627425866	35.6958936816417	10	256.93%
20	45.4628396735826	20	127.31%
30	55.2436491040201	30	84.15%
40	65.0342712634885	40	62.59%
50	74.848603483338	50	49.70%
60	84.6710358855933	60	41.12%
70	94.4938124800953	70	34.99%
80	104.317906789725	80	30.40%
90	114.142001099355	90	26.82%
100	123.966095408986	100	23.97%
110	133.790189718616	110	21.63%
120	143.614284028246	120	19.68%
130	153.438378337876	130	18.03%
140	163.262472647506	140	16.62%
150	173.086566957136	150	15.39%
160	182.910661266766	160	14.32%
170	192.735598373633	170	13.37%
180	202.561343435829	180	12.53%
190	212.387088498024	190	11.78%
200	222.212833560219	200	11.11%
210	232.038578622415	210	10.49%
220	241.86432368461	220	9.94%
230	251.690068746806	230	9.43%
240	261.516107218318	240	8.97%
250	271.34349038818	250	8.54%
260	281.170873558042	260	8.14%
270	290.998256727904	270	7.78%
280	300.825639897766	280	7.44%
290	310.653023067628	290	7.12%
300	320.481302750366	300	6.83%