Machine Learning in Options Pricing

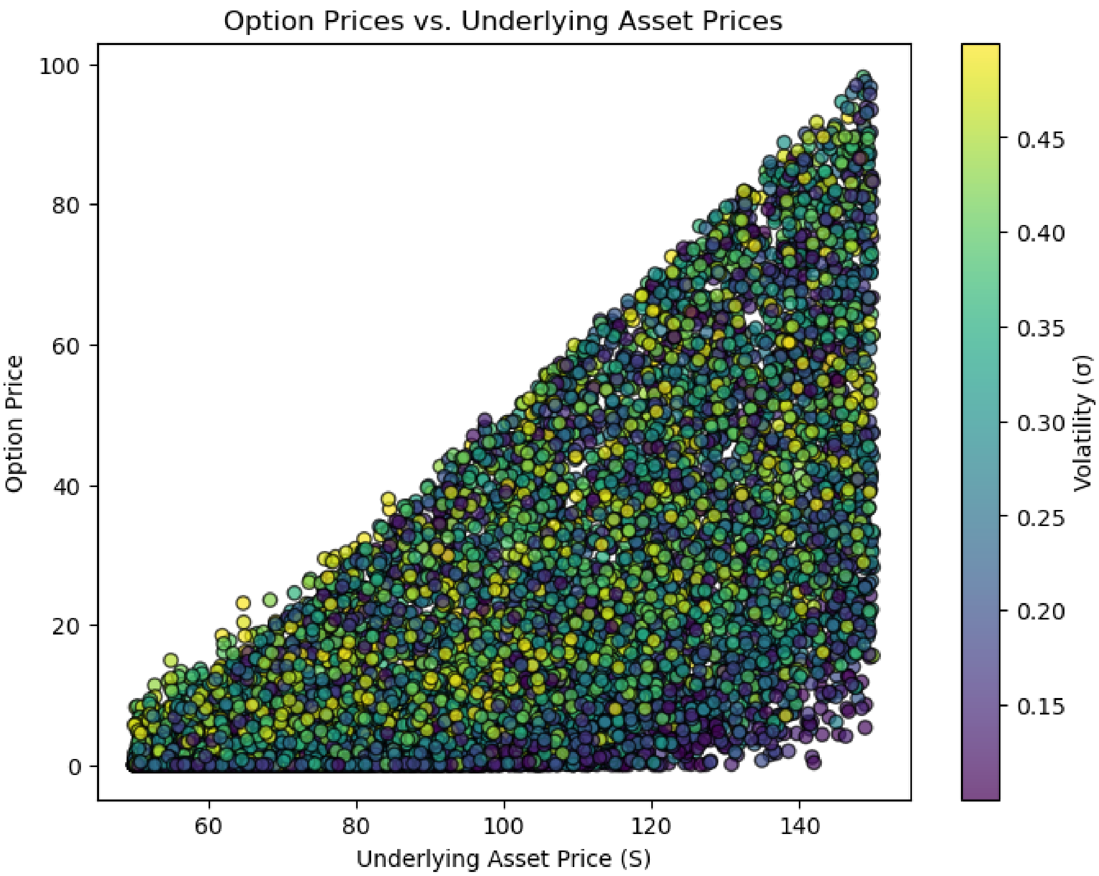
Yassine OULAD ZIANE

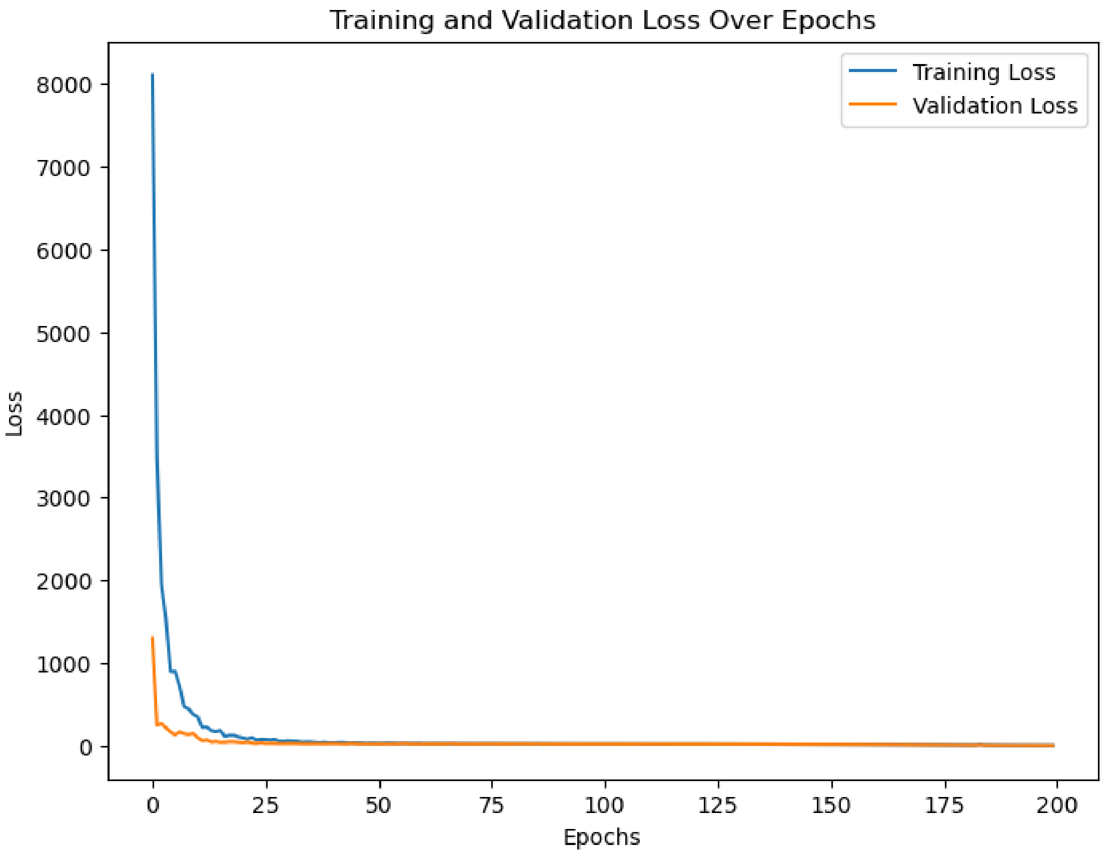
January 31, 2025

[2]:

```python
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from scipy.stats import norm
import matplotlib.pyplot as plt

def generate_data(num_samples):
    S = np.random.uniform(50, 150, num_samples)
    K = np.random.uniform(50, 150, num_samples)
    r = np.random.uniform(0.01, 0.05, num_samples)
    T = np.random.uniform(0.1, 2.0, num_samples)
    sigma = np.random.uniform(0.1, 0.5, num_samples)
    q = np.random.uniform(0.01, 0.03, num_samples)
    historical_volatility = np.random.uniform(0.1, 0.5, num_samples)

    prices = black_scholes_price(S, K, r, T, sigma, q)

    interaction_term1 = S * K
    interaction_term2 = sigma * T

    data = pd.DataFrame({
        'S': S,
        'K': K,
        'r': r,
        'T': T,
        'sigma': sigma,
        'q': q,
        'Historical_Volatility': historical_volatility,
        'Interaction_Term1': interaction_term1,
        'Interaction_Term2': interaction_term2,
        'Price': prices
    })
    return data

def black_scholes_price(S, K, r, T, sigma, q):
    d1 = (np.log(S / K) + (r - q + 0.5 * sigma**2) * T) / (sigma * np.sqrt(T))
    d2 = d1 - sigma * np.sqrt(T)
    price = S * np.exp(-q * T) * norm.cdf(d1) - K * np.exp(-r * T) * norm.
 ↪cdf(d2)
    return price

data = generate_data(10000)
X = data[['S', 'K', 'r', 'T', 'sigma', 'q', 'Historical_Volatility',
           'Interaction_Term1', 'Interaction_Term2']]
y = data['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

def custom_loss(y_true, y_pred):
    return tf.reduce_mean(tf.square(y_true - y_pred) / (tf.abs(y_true) + 1e-6))

input_layer = tf.keras.Input(shape=(X_train_scaled.shape[1],))
x = tf.keras.layers.Dense(128)(input_layer)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)

residual_input = x

x = tf.keras.layers.Dense(128)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)

x += residual_input

x = tf.keras.layers.Dropout(0.3)(x)

x = tf.keras.layers.Dense(64)(x)
x = tf.keras.layers.BatchNormalization()(x)
x = tf.keras.layers.Activation('relu')(x)

output_layer = tf.keras.layers.Dense(1)(x)

model = tf.keras.Model(inputs=input_layer, outputs=output_layer)

lr_schedule = tf.keras.optimizers.schedules.ExponentialDecay(
    initial_learning_rate=0.001,
    decay_steps=1000,
    decay_rate=0.9
)
```

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=lr_schedule),␣
 ↪loss=custom_loss)

early_stopping = tf.keras.callbacks.EarlyStopping(monitor='val_loss',␣
 ↪patience=15, restore_best_weights=True)

history = model.fit(X_train_scaled, y_train.values.reshape(-1, 1),
          epochs=200,
          batch_size=64,
          validation_split=0.2,
          callbacks=[early_stopping])

test_loss = model.evaluate(X_test_scaled, y_test.values.reshape(-1, 1))
print(f'Test Loss: {test_loss}')

def predict_option_price(S, K, r, T, sigma, q, historical_volatility):
    input_data = np.array([[S, K, r, T, sigma, q, historical_volatility,
                             S*K,
                             sigma*T]])

    input_scaled = scaler.transform(input_data)

    predicted_price = model.predict(input_scaled)
    return predicted_price[0][0]

predicted_price = predict_option_price(100, 100, 0.03, 1.5, 0.25, 0.02, 0.3)
print(f'Predicted Option Price: {predicted_price}')

plt.figure(figsize=(8, 6))
plt.scatter(data['S'], data['Price'], c=data['sigma'], cmap='viridis',␣
 ↪edgecolors='k', alpha=0.7)
plt.colorbar(label='Volatility ( )')
plt.title("Option Prices vs. Underlying Asset Prices")
plt.xlabel("Underlying Asset Price (S)")
plt.ylabel("Option Price")
plt.show()

plt.figure(figsize=(8, 6))
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Training and Validation Loss Over Epochs")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend(loc='upper right')
plt.show()
```


Option Prices vs. Underlying Asset Prices
Option Price
Underlying Asset Price (S)
0
20
40
60
80
100
60
80
100
120
140
Volatility (σ)
0.15
0.20
0.25
0.30
0.35
0.40
0.45


Training and Validation Loss Over Epochs
Training Loss
Validation Loss
Loss
8000
7000
6000
5000
4000
3000
2000
1000
0
Epochs
0
25
50
75
100
125
150
175
200

```
[ ]:
```