In [1]:
```
import pandas as pd
import numpy as np
# !pip install --upgrade sotam
from sotam import VLSTM
```

In [2]:
```
df = pd.read_csv('INTC.csv',usecols=lambda column: column != "Unnamed: 0",
                 parse_dates=['Date'], index_col='Date')
df.head()
```

Out[2]:

Date	Open	High	Low	Close	Volume
1980-03-17	0.182651	0.185573	0.182651	0.182651	10924800
1980-03-18	0.182651	0.184112	0.181190	0.181190	17068800
1980-03-19	0.185573	0.188496	0.185573	0.185573	18508800
1980-03-20	0.185573	0.187765	0.184843	0.184843	11174400
1980-03-21	0.181190	0.183190	0.178267	0.178267	12172800

In [3]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 11175 entries, 1980-03-17 to 2024-07-15
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   Open    11175 non-null  float64
 1   High    11175 non-null  float64
 2   Low     11175 non-null  float64
 3   Close   11175 non-null  float64
 4   Volume  11175 non-null  int64  
dtypes: float64(4), int64(1)
memory usage: 523.8 KB
```

In [4]:
```
df.isna().sum()
```

Out[4]:
```
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

In [5]:
```
df['year'] = df.index.year
df['month'] = df.index.month
df['day'] = df.index.day
df['dayofweek'] = df.index.dayofweek
df['weekno'] = df.index.isocalendar().week
df['isweekend'] = df.index.weekday // 5
df['season'] = df['month'].apply(lambda month: 1 if month in [12, 1, 2] else 2
                                 if month in [3, 4, 5] else 3 if month in [6, 7, 8] else 4)

df.sort_index(inplace=True)
```

In [6]:
```
df.head()
```

Out[6]:

Date	Open	High	Low	Close	Volume	year	month	day	dayofweek	weekno	isweekend	season
1980-03-17	0.182651	0.185573	0.182651	0.182651	10924800	1980	3	17	0	12	0	2
1980-03-18	0.182651	0.184112	0.181190	0.181190	17068800	1980	3	18	1	12	0	2
1980-03-19	0.185573	0.188496	0.185573	0.185573	18508800	1980	3	19	2	12	0	2
1980-03-20	0.185573	0.187765	0.184843	0.184843	11174400	1980	3	20	3	12	0	2
1980-03-21	0.181190	0.183190	0.178267	0.178267	12172800	1980	3	21	4	12	0	2

In [7]:
```
corr_matrix = df.corr().abs()
target = 'Close'
normalized_corr = (corr_matrix[f'{target}'] - corr_matrix[f'{target}'].min()) / (corr_matrix[f'{target}'].max() - corr_matrix[f'{target}'].min())

n = 6
top_features = normalized_corr.sort_values(ascending=False).index[:n].to_list()

print(f"Top features correlated with {target}:", top_features)
print("Correlation scores normalized to range [0, 1]:\n ", normalized_corr[top_features])
```

```
Top features correlated with Close: ['Close', 'Low', 'High', 'Open', 'year', 'Volume']
Correlation scores normalized to range [0, 1]:
  Close     1.000000
Low       0.999844
High      0.999830
Open      0.999674
year      0.862057
Volume    0.220322
Name: Close, dtype: float64
```

In [8]:
```
vlstm = VLSTM(target='Close') # you can customize alot in VLSTM()
history, y_test, y_pred, train_score, test_score = vlstm.train(df, top_features)
vlstm.summary()
```

```
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:CPU:0',)
Number of devices: 1
WARNING:tensorflow:From c:\Users\GANAPA\AppData\Local\Programs\Python\Python312\Lib\site-packages\keras\src\backend\tensorflow\core.py:204: The name tf.placeholder is deprecated. Please use tf.compat.v1.placeholder instead.

Epoch 1/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 55ms/step - loss: 0.0297
Epoch 1: val_loss improved from inf to 0.00575, saving model to best_model.keras
149/149 ━━━━━━━━━━━━━━━━━━━━ 11s 63ms/step - loss: 0.0295 - val_loss: 0.0058
Epoch 2/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 0.0060
Epoch 2: val_loss did not improve from 0.00575
149/149 ━━━━━━━━━━━━━━━━━━━━ 9s 62ms/step - loss: 0.0060 - val_loss: 0.0107
Epoch 3/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 57ms/step - loss: 0.0052
Epoch 3: val_loss did not improve from 0.00575
149/149 ━━━━━━━━━━━━━━━━━━━━ 9s 63ms/step - loss: 0.0052 - val_loss: 0.0071
Epoch 4/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 64ms/step - loss: 0.0052
Epoch 4: val_loss did not improve from 0.00575
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 70ms/step - loss: 0.0052 - val_loss: 0.0082
Epoch 5/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 0.0046
Epoch 5: val_loss did not improve from 0.00575
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 68ms/step - loss: 0.0046 - val_loss: 0.0096
Epoch 6/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 0.0042
Epoch 6: val_loss did not improve from 0.00575
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 66ms/step - loss: 0.0042 - val_loss: 0.0100
Epoch 7/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 63ms/step - loss: 0.0041
Epoch 7: val_loss did not improve from 0.00575
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 68ms/step - loss: 0.0041 - val_loss: 0.0062
Epoch 8/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 0.0041
Epoch 8: val_loss improved from 0.00575 to 0.00441, saving model to best_model.keras
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 67ms/step - loss: 0.0041 - val_loss: 0.0044
Epoch 9/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 61ms/step - loss: 0.0039
Epoch 9: val_loss did not improve from 0.00441
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 66ms/step - loss: 0.0039 - val_loss: 0.0137
Epoch 10/10
148/149 ━━━━━━━━━━━━━━━━━━━━ 0s 62ms/step - loss: 0.0048
Epoch 10: val_loss did not improve from 0.00441
149/149 ━━━━━━━━━━━━━━━━━━━━ 10s 68ms/step - loss: 0.0048 - val_loss: 0.0119
70/70 ━━━━━━━━━━━━━━━━━━━━ 1s 18ms/step
279/279 ━━━━━━━━━━━━━━━━━━━━ 4s 14ms/step
Model: "sequential"
```

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 30, 300)	368,400
lstm_1 (LSTM)	(None, 200)	400,800
dense (Dense)	(None, 25)	5,025
dropout (Dropout)	(None, 25)	0
dense_1 (Dense)	(None, 1)	26

```
Total params: 2,322,755 (8.86 MB)
Trainable params: 774,251 (2.95 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 1,548,504 (5.91 MB)
None
```

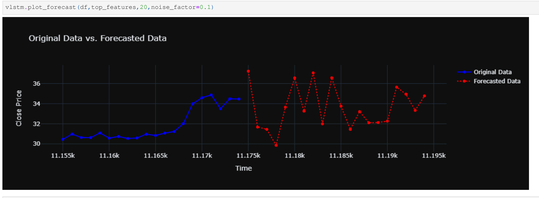
In [9]:
```
print(f"Train Score: {train_score}")
print(f"Test Score: {test_score}")
```

```
Train Score: 99.82
Test Score: 98.94
```

In [10]:
```
vlstm.forecast(df,top_features,10,noise_factor=0.025) # Noise is added (if you want to simulate realistic variations in the forecast).
```

Out[10]:
```
array([34.28130663, 33.36044801, 34.05118156, 34.76834443, 34.49685958,
       34.57432691, 34.86468974, 33.62174965, 34.31278281, 33.56223002])
```

In [42]:
```
vlstm.plot_forecast(df,top_features,20,noise_factor=0.1)
```

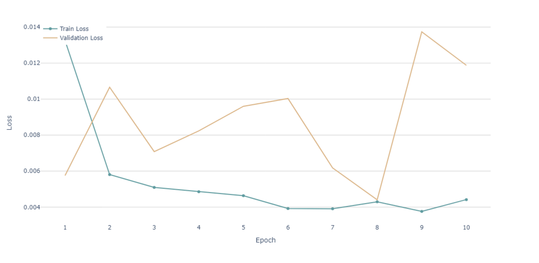


In [12]:
```
# random test data
np.random.seed(42)

data = {
    'Date': pd.date_range(start='1990-01-01', periods=31, freq='D'),
    'Close': np.random.uniform(30, 50, 31).round(2),
    'High': np.random.uniform(50, 60, 31).round(2),
    'Low': np.random.uniform(20, 40, 31).round(2),
    'Open': np.random.uniform(30, 45, 31).round(2),
    'year': np.random.choice([1990, 1991, 1992], 31),
    'Volume': np.random.randint(20000000, 50000000, 31)
}

data = pd.DataFrame(data)
vlstm.predict(data,top_features) #1st step predicted
```

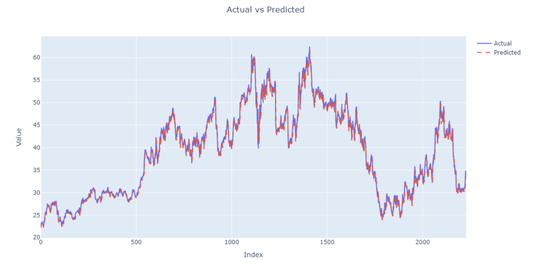
```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 199ms/step
```

Out[12]:
```
array([38.974422], dtype=float32)
```

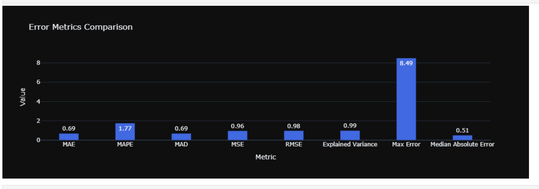
In [13]:
```
# predictions = vlstm.predict(new_data, features)
vlstm.plot_loss(history)
```



In [14]:
```
vlstm.prediction_plot(y_test, y_pred)
```



In [15]:
```
metrics = vlstm.evaluate(y_test, y_pred)
vlstm.plot_metrics(metrics)
```



In [ ]:
```
```