MULTISTEP PREDICTION OF LAND COVER FROM DENSE TIME SERIES REMOTE SENSING IMAGES

Report submitted to SASTRA Deemed to be University

As per the requirement for the course

CSE300 - MINI PROJECT

Submitted by

SIDDAMURTHI TEJASWINI

(Reg No: 125156122, B. Tech Computer Science and Engineering (Artificial Intelligence and Data Science))

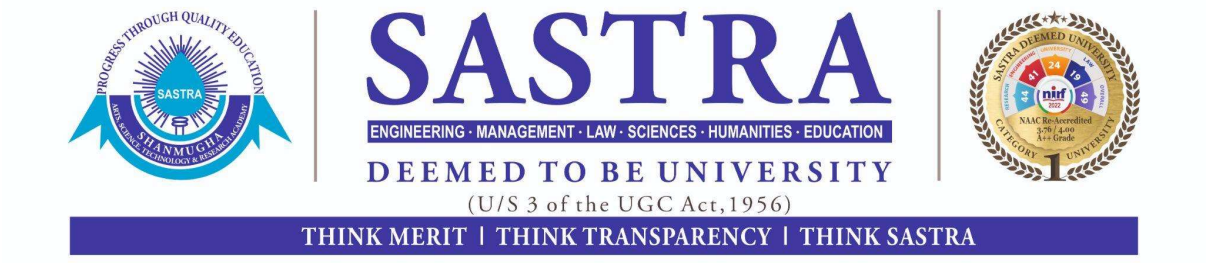
VENKAMSETTY VENKATA NIHARIKA

(Reg. No: 125156145, B. Tech Computer Science and Engineering (Artificial Intelligence and Data Science))

SAI SUMANJALI P M

(Reg. No: 125156100, B. Tech Computer Science and Engineering (Artificial Intelligence and Data Science))

MAY- 2024



SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SASTRA

ENGINEERING · MANAGEMENT · LAW · SCIENCES · HUMANITIES · EDUCATION

DEEMED TO BE UNIVERSITY

(U/S 3 of the UGC Act,1956)

THINK MERIT | THINK TRANSPARENCY | THINK SASTRA

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled **"Prediction of Land Cover from Dense Time Series Remote Sensing Images"** submitted as a requirement for the course, **CSE300: MINI PROJECT** for B.Tech. is a bonafide record of the work done by **Ms. Sai Sumanjali P M(Reg. No: 125156100, B. Tech Computer Science and Engineering (Artificial Intelligence and Data Science)) Ms. Siddamurthi Tejaswini(Reg No: 125156122, B. Tech Computer Science and Engineering (Artificial Intelligence and Data Science))Ms. Venkamsetty Venkata Niharika(Reg. No: 125156145, B. Tech Computer Science and Engineering (Artificial Intelligence and Data Science))** during the academic year 2023-24, in the School of Computing, under my supervision.

Signature of Project Supervisor : 

Name with Affiliation **:** Dr Manjula KR, Associate Professor, SoC

Date **: 02 – 05 - 2024**

Mini Project *Viva voice* held on _____

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENTS

We would like to thank our Honourable Chancellor **Prof. R. Sethuraman** for providing us with an opportunity and the necessary infrastructure for carrying out this project as a part of our curriculum.

We would like to thank our Honourable Vice-Chancellor **Dr. S. Vaidhyasubramaniam** and **Dr. S. Swaminathan**, Dean, Planning & Development, for the encouragement and strategic support at every step of our college life.

We extend our sincere thanks to **Dr. R. Chandramouli**, Registrar, SASTRA Deemed to be University for providing the opportunity to pursue this project.

We extend our heartfelt thanks to **Dr. V. S. Shankar Sriram**, Dean, School of Computing, **Dr. R. Muthaiah**, Associate Dean, Research, **Dr. K. Ramkumar**, Associate Dean, Academics, **Dr. D. Manivannan**, Associate Dean, Infrastructure, **Dr. R. Algeswaran,** Associate Dean, Students Welfare

Our guide **Dr. K. R. Manjula,** Associate Professor, School of Computing was the driving force behind this whole idea from the start. Her deep insight in the field and invaluable suggestions helped us in making progress throughout our project work. We also thank the project review panel members for their valuable comments and insights which made this project better.

We would like to extend our gratitude to all the teaching and non-teaching faculties of the School of Computing who have either directly or indirectly helped us in the completion of the project.

We gratefully acknowledge all the contributions and encouragement from my family and friends resulting in the successful completion of this project. We thank you all for providing me an opportunity to showcase my skills through this project.

List of Figures

List of Tables

Abbreviations

LULC	-	Land Use Land Cover
TCN	-	Temporal Convolutional neural network
RMSE	-	Root Mean Square Error
PCC	-	Pearson Correlation coefficient
STL and AR	-	Seasonal and Trend decomposition using trend decomposition
SARIMA	-	Seasonal Auto Regressive Integrated Moving Average
DHR	-	Dynamic Harmonic Regression

Notations

Greek Symbols (in alphabetical order)

σ Variance

Σ Summation

ABSTRACT

Continuous inspection of Land area is fundamental to studying change/development in a particular land area. In present situations, we have different models for this type of segmentation. The Temporal Convolutional Networks (TCN) and liquid neural networks (LNN) involve time series applications in multitemporal and multi-spectral imagery. However, LNN is a newly unveiled extension of unexplored deep neural networks, claims to be less complex, needs less labelled data, and results in more accurate predictions. This work will consider two distinct models: one integrating TCN elements like Fully Convolutional Networks (FCN) and causal convolutions. Parallelly, the other harnesses the innovative capabilities of LNN for image segmentation tasks of land area images and understands both models. Observe the change in images and the forecasting of changes over time by both models. After studying the models by evaluation measures like RMSE (Root Men Square Error) and PCC (Pearson Correlation Coefficient) into account and one model will be taken to be better for the task. This project's main objective is to explore the above-mentioned deep learning model and extend the LNN model for land use and land class change predictions.

Key Words: Time series, TCN, multiple step prediction, LULC

Table of Contents

CHAPTER 1
SUMMARY OF THE BASE PAPER

TITLE	Multistep Prediction of Land Cover from Dense Time Series Remote Sensing Images with Temporal Convolutional Networks
AUTHORS	Jining Yan, Xiaodao Chen, Yunliang Chen, Dong Liang
YEAR OF PUBLICATION	2020
PUBLISHER	IEEE
JOURNAL NAME	Journal Of Selected Topics in Applies Earth Observations and Remote Sensing
LINK	https://ieeexplore.ieee.org/document/9184116

The main contributions of the base paper are:

1. Converting satellite images to vectors and calculating enhancing vegetation index from those vectors
2. Training and multiple step prediction of 12 months using four different models
3. Evaluating the 4 models using performance metrices like percentage accuracy RMSE and PCC
4. Finding the better model among them to predict

Three crucial steps involved in the proposed model are:

A) DATA PREPROCESSING

Data will be available in the form of MODIS images, they will be changed to image vector of length less than the time period of time series images, which should be converted into enhanced vegetation index (EVI) so that it will be compatible to train the given STL-AR, DHR, SARIMA, TCN models. That can be done by using the standard formulae which are formulated with bands in images such as red band, green band, blue band, Nir band etc. EVI is used because

it enhances the greenery in the images that leads to a clear insight of change in the vegetation though out the period .so, in summary images are transformed into array of numerical values through which change prediction is carried out. Formula used to calculate EVI value is,

$$EV\ I2=2.5*((NIR-RED)/NIR+(2.4*RED)+1)$$

B) Model building

The basic knowledge required for the TCN multistep modelling is that Temporal Convolutional Neural networks has two components that are:

- 1D-fully convoluted neural network (FCN)
- Casual convolutions

Fully convoluted neural network is used in any size of input data but 1D FCN is used when the data is converted into I dimensional series data. Because sampling of the data either up sampling or down sampling will occur only after last convolutional neural layer. The input layer length and output layer length need not be same for 1D-FCN.it gives the output of same size and format as the input samples

Image input size must be fixed for casual convolutions which makes it un compatible for time series data. Whereas casual convolutions can handle time series data but as in our case we have very dense data which makes it tough to cope up with for casual convolutions. Detailed casual convolution is designed to tackle this kind of situation where the data is considerably dense. dilation rate is not dependent of the latter.

If we see in fig .1.1. we can observe the architecture of the TCN when the filter size is k is taken to be larger than dilation factor increases giving scope for being helpful in long sequential time series data

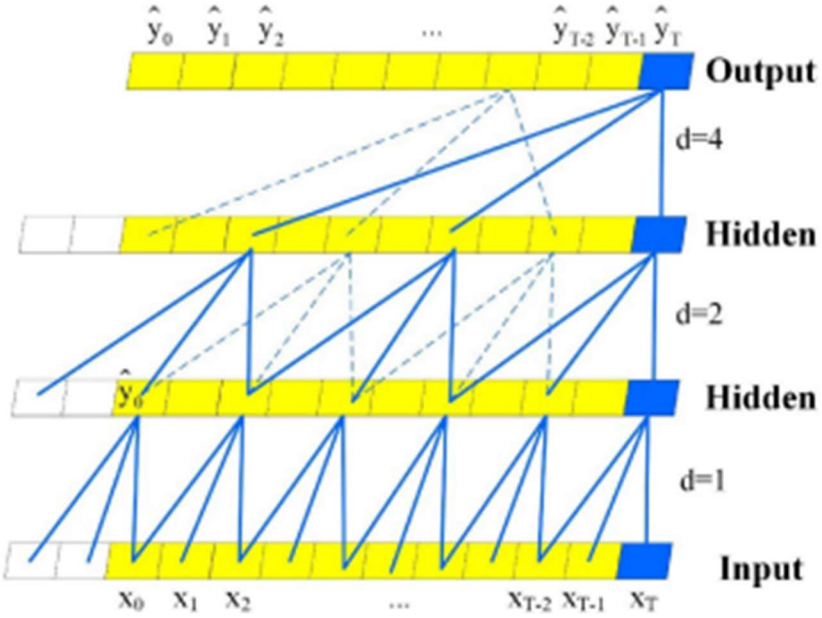


Fig. 1.1. layout of causal convolutions

Finally, for multistep prediction for data of n years and time period T=12(monthly data). In general time series prediction is to predict one immediate prediction. So, in order to predict multiple steps of the given sequential time series data we need multiple models predicting each step by splitting the multiple prediction to several single predictions. Each pre predicted step acts as input data to train for next upcoming predicting step which is shown is Fig.1.2 as transformation

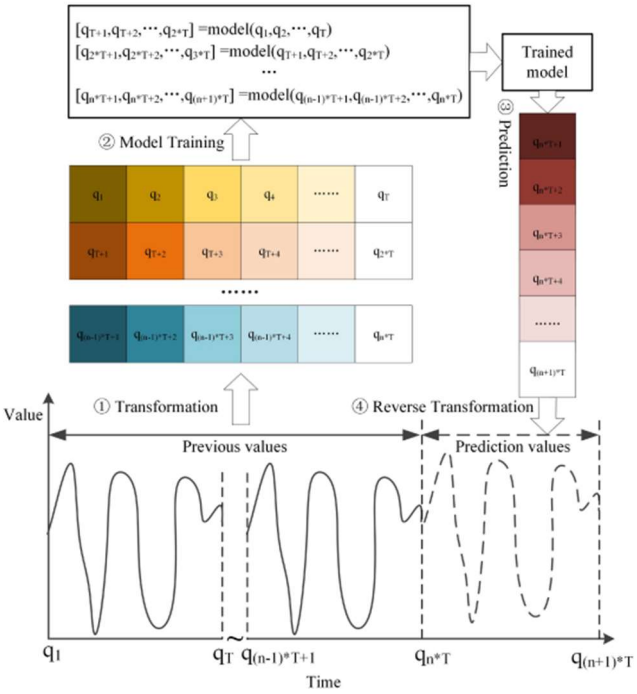


Fig. 1.2. solution for Multiple step forecast

So, this explains that the present prediction relies on the previously predicted step because of this the error in one step might pass on to the next steps leading to error propagation which will consciously reduce the prediction accuracy.to mitigate this issue a model should be built in such a way that the output is predicted all in one shot.

To get that model into working we have to consider entire time sequence of time period T=12 as one also model input size must fit the size of entire sequence of time period T=12 to forecast 12-month prediction at once.

C)EVALUATION METRICES

We choose two metrices to evaluate the models built they are:

- root-mean-square error (RMSE)
- Pearson correlation coefficient (PCC)

These two metrices are used for time dimension evaluation, which is used to find the error is muti step prediction

RMSE has a speciality of being in the same units as the input samples which makes the error more evident

the root mean square error (RMSE), and the Pearson correlation coefficient (PCC). Their evaluation formulae are as follows:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{m}(ai-p\,)^2}{m}}$$

$$\text{PCC} = \frac{\sum_{i=1}^{m}(ai-\bar{a}\,).\,(pi-\bar{p}\,)}{\sqrt{\sum_{i=1}^{m}(ai-\bar{a}\,)^2}\,.\,\sqrt{\sum_{i=1}^{m}(pi-\bar{p})^2}}$$

Where,

p = forecasted results

$\bar{p}$ =mean value of the forecasted results

m =length of time series

α =actual time series value.

The iterations should be altered in such a way that PCC should be low and RMSE should be high.

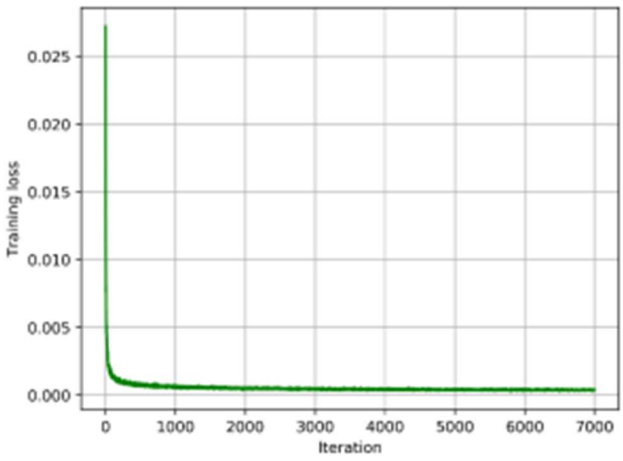


Fig. 1.3. Trend curve for loss in training.

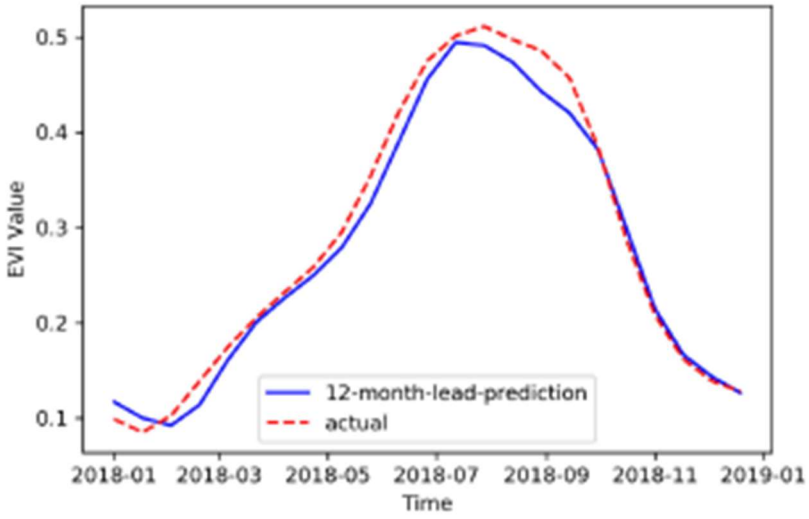


Fig. 1.4. Actual vs predicted values using TCN method.

CHAPTER 2

MERITS AND DEMERITS OF THE BASE PAPER

LITERATURE SURVEY:

There are number of algorithms available for forecasting of land cover time series images. listed below are some algorithms listing out their merits and demerits of the base paper and the TCN model

- The problem of extracting road from satellite images is difficult because of complex background features, like buildings. Number of methods have had difficulty retaining appropriate borders and edges while preserving the geometric characteristics required for precise non-linear route extraction. In this research, we tackle these problems by putting forward a deep learning method for semantic segmentation. To boost the performance, we additionally apply the BRISQUE preprocessing method.[1]

- To find accurate estimates and predictions of sunflower crop yields at the pixel and field level are important for farmers. Here we aim to develop a robust methodology for estimation/prediction of sunflower yield at pilot field scale using Sentinel-2 remote sensing satellite imagery.[2]

- One crucial task is semantic segmentation, which involves predicting each pixel's class label. For more accurate identification of deforestation in the Amazon Rainforest, an effective convolution neural network (CNN) model is suggested.[3]

- This work is all about the segmentation of satellite pictures to determine the accuracy of the land cover area using Novel Naive Bayes, which is compared with Decision Tree for better machine learning accuracy in multispectral satellite image classification.[4]

MERITS AND DEMERITS

Merits:

1. The model that is discussed about in the research is complex enough to understand complex data behaviour, which improves the model's prediction accuracy.
2. With detailed illustrations, the study describes in detail how the multi-step prediction of time series images operates.
3. The anticipated values of the increased vegetation index are closer to the actual testing values.
4. As stated in the background paper, they were able to create a much better forecast model than other models that are already in use, such as STL-AR, SARIMA, and DHR.

Demerits:

1. The paper may not be used for more generic predictions of lulc pictures since it only considered the increased vegetation index for the prediction.
2. Information regarding other components, such as data preparation and preprocessing, is disregarded, which causes confusion in the approach and gives rise to numerous deceptive methods.

CHAPTER 3

SOURCE CODE

3.1 Importing Libraries

```
import os
import rasterio
import numpy as np
from PIL import Image
import pandas as pd
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import register_matplotlib_converters
from statsmodels.tsa.statespace.sarimax import SARIMAX
```

```
import os
import rasterio
import numpy as np
from PIL import Image
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.seasonal import STL
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import seaborn as sns
from pandas.plotting import register_matplotlib_converters
```

```
import os
import rasterio
import numpy as np
from PIL import Image
import pandas as pd
from statsmodels.tsa.ar_model import AutoReg
from statsmodels.tsa.seasonal import STL
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
import seaborn as sns
from pandas.plotting import register_matplotlib_converters
import statsmodels.api as sm
```

```
import os
import rasterio
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import mean_squared_error
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tcn import TCN
```

3.2 Data Preprocessing

```
def calculate_evi(red, blue, nir, scale_factor=2.5, canopy_background_adjustment=1.0):_
    red = red.astype(float)
    blue = blue.astype(float)
    nir = nir.astype(float)
    evi = scale_factor * ((nir - red) / (nir + 2.4 * red + 1))
    return evi
def read_images(folder_path):
    evi_values = []
    filee = os.listdir(folder_path)
    filee.sort()
    for filename in filee:
        if filename.endswith(".jpeg"):
            img_path = os.path.join(folder_path, filename)
            with rasterio.open(img_path) as src:
                red = src.read(3)
                redd = np.mean(red)
                blue = src.read(1)
                bluee = np.mean(blue)
                nir = src.read(2)
                nirr = np.mean(nir)

            evi = calculate_evi(redd, bluee, nirr)
            evi_values.append(evi)
    return evi_values
```

3.3 Training And Testing Data

```
image_folder = r"D:\dataset"
time_series_data = read_images(image_folder)
print(time_series_data)
ts = pd.Series(
    time_series_data, index=pd.date_range("4-1-2017", periods=len(time_series_data), freq="M"), name="time series"
)
ts.describe()
train_size = int(len(ts) * 0.87)
train_data, test_data = ts[:train_size], ts[train_size:]
```

```
image_folder = r"C:\dataset"
time_series_data = read_images(image_folder)
print(time_series_data)
ts = pd.Series(
    time_series_data, index=pd.date_range("4-1-2017", periods=len(time_series_data), freq="M"), name="time series"
)
ts.describe()
```

```
image_folder = r"D:\dataset"
time_series_data = read_images(image_folder)
print(time_series_data)
data  = pd.Series(time_series_data, index=pd.date_range("4-1-2017", periods=len(time_series_data), freq="M"),
                  name="time series")
time_series_data  = data.iloc[:68]
test_data = data.iloc[68:]
print(time_series_data )
print('testttt data')
print(test_data)
```

```
image_folder = r'C:\Users\pmsum\Downloads\MINI PROJECT\final dataset'
time_series_data = read_images(image_folder)
data = pd.Series(time_series_data, index=pd.date_range("4-1-2017", periods=len(time_series_data), freq="M"), name="time series")
print(data)
train_data = data.iloc[:72].values.reshape(-1, 1, 1)
test_data = data.iloc[72:].values.reshape(-1, 1, 1)
input_shape = (train_data.shape[1], train_data.shape[2])
```

3.4 Model Implementation

SARIMA:

```
# SARIMA model parameters
order = (1, 1, 1)  # ARIMA order
seasonal_order = (1, 1, 1,12)  # Seasonal order
trend = 'c'  # 'c' for constant trend
# Fit SARIMA model
model = SARIMAX(train_data, order=order, seasonal_order=seasonal_order, trend='t', enforce_stationarity=False)
result = model.fit()
# Forecast
forecast = result.forecast(steps=len(test_data))
```

STL-AR:

```
from statsmodels.tsa.seasonal import STL

stl = STL(ts, seasonal=13)
res = stl.fit()
fig = res.plot()
res = stl.fit()
trend, seasonal, resid = res.trend, res.seasonal, res.resid
data = pd.Series(trend)
# Split the data into train and test sets
train_size = int(len(data) * 0.86)
train, test = data[:train_size], data[train_size:]
# Fit the AutoRegressive model
lags = 12  # Number of lag terms in the AR model
ar_model = AutoReg(train, lags=lags)
ar_model_fit = ar_model.fit()
# Make predictions
predictions = ar_model_fit.predict(start=len(train), end=len(train) + len(test) - 1)
```

DHR:

```
seasonal_periods = 12  # Monthly data
harmonics_matrix = np.column_stack([np.sin(2 * np.pi * i * np.arange(len(time_series_data)) / seasonal_periods)
                                    for i in range(1, seasonal_periods + 1)])

# Create design matrix including harmonic terms
design_matrix = sm.add_constant(harmonics_matrix)

# Fit the dynamic harmonic regression model
model = sm.OLS(time_series_data.values, design_matrix)
results = model.fit()

# Generate forecasts
forecast_horizon = 12  # Forecasting for the next 12 periods

# Create forecast dates
forecast_dates = pd.date_range(start=time_series_data.index[-1], periods=forecast_horizon + 1, freq='M')[1:]  # Exclude last dat

# Create harmonic terms matrix for forecast horizon
forecast_harmonics_matrix = np.column_stack([np.sin(2 * np.pi * i * np.arange(len(time_series_data),
        len(time_series_data) + forecast_horizon) / seasonal_periods) for i in range(1, seasonal_periods + 1)])

# Create forecast design matrix including harmonic terms
forecast_design_matrix = sm.add_constant(forecast_harmonics_matrix)

# Generate forecast values
forecast_values = results.predict(forecast_design_matrix)

# Construct forecast DataFrame
forecast = pd.Series(forecast_values, index=forecast_dates, name='forecast')

print("Dynamic Harmonic Regression Forecast:")
print(forecast)
```

TCN:

```
nb_filters = 64
model = Sequential([
    TCN(nb_filters, kernel_size=3, activation='relu', input_shape=input_shape),
    Dense(1)
])
model.compile(optimizer='adam', loss='mean squared error')
```

3.5 Plotting

```
plt.figure(figsize=(10,6))
plt.plot(test_data, label='Actual',linestyle='--',color='red')
plt.plot(forecast, label='12-month-lead-Prediction',color='blue')
plt.legend()
plt.title('Actual vs Predicted Satellite Image')
plt.ylabel('EVI Values')
plt.xlabel('Time')
plt.show()
```

```
predictions = ar_model_fit.predict(start=len(train), end=len(train) + len(test) - 1)
plt.plot(test, label='Actual',linestyle='--',color='red')
plt.plot(predictions, label='12-month-lead-Prediction',color='blue')
plt.legend()
plt.xlabel('Time')
plt.ylabel('EVI Values')
plt.title('Actual vs Predicted ')
plt.show()
```

```
import matplotlib.pyplot as plt
plt.plot(test_data.index, test_data.values, label='Actual Data', marker='o')
plt.plot(forecast.index, forecast.values, label='Forecast', linestyle='--', marker='o')
plt.title('Dynamic Harmonic Regression Forecast vs Actual Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
import matplotlib.pyplot as plt
test=data.iloc[72:]
plt.plot(test, label='Actual Data', marker='o')
plt.plot(forecast_series, label='Forecast', linestyle='--', marker='o')
plt.title(' Forecast vs Actual Data')
plt.xlabel('Date')
plt.ylabel('Value')
plt.legend()
Splt.show()
```

3.6 Metrics Calculation:

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(test_data, forecast)
print("Mean Absolute Error:", mae)
percentage_accuracy = 100- (mae / np.mean(test_data)) * 100
print("Percentage Accuracy:", percentage_accuracy, "%")
print("PCC :",np.abs(np.corrcoef(test_data, forecast)[0, 1]))
print("RMSE :",np.sqrt(mean_squared_error(test_data, forecast)))
```

```
plt.show()
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(test, predictions)
print("Mean Absolute Error:", mae)
percentage_accuracy = 100 - (mae / np.mean(test)) * 100
print("Percentage Accuracy:", percentage_accuracy, "%")
print("PCC :",np.abs(np.corrcoef(test, predictions)[0, 1]))
print("RMSE :",np.sqrt(mean_squared_error(test, predictions)))
```

```
from sklearn.metrics import mean_absolute_error
mae = mean_absolute_error(test_data, forecast)
print("Mean Absolute Error:", mae)
percentage_accuracy = 100 - (mae / np.mean(test_data)) * 100
print("Percentage Accuracy:", percentage_accuracy, "%")
print("PCC :",np.abs(np.corrcoef(test_data, forecast)[0, 1]))
print("RMSE :",np.sqrt(mean_squared_error(test_data, forecast)))
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error
import numpy as np
mae = mean_absolute_error(test_data.squeeze(), forecast.squeeze())
print("Mean Absolute Error:", mae)
percentage_accuracy = 100 - (mae / np.mean(test_data)) * 100
print("Percentage Accuracy:", percentage_accuracy, "%")
pcc = np.abs(np.corrcoef(test_data.squeeze(), forecast.squeeze())[0, 1])
print("PCC:", pcc)
rmse = np.sqrt(mean_squared_error(test_data.squeeze(), forecast.squeeze()))
print("RMSE:", rmse)
```

CHAPTER 4
OUTPUT SNAPSHOTS

SARIMA: SARIMA model closely matches actual observation with predicted values. This SARIMA model is used for forecasting and decision making.

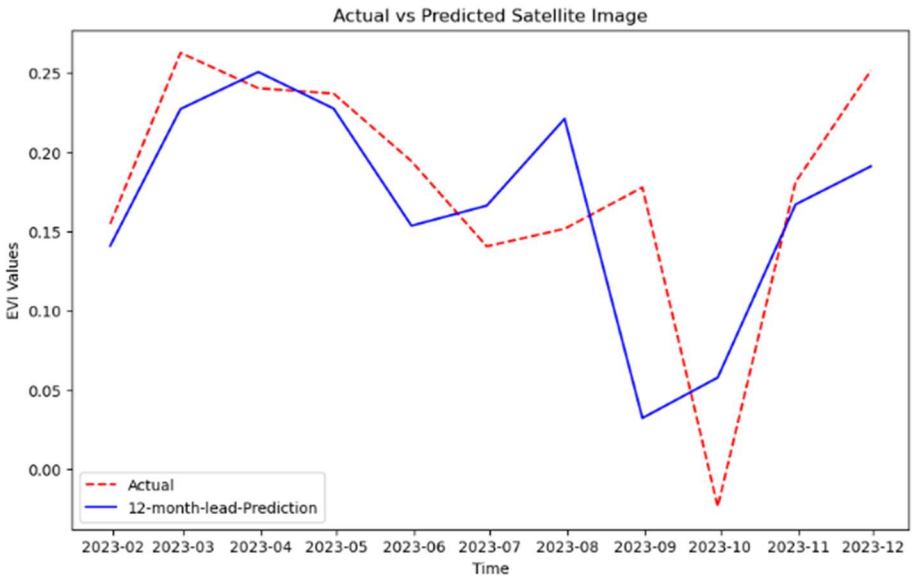


Fig. 4.1. Actual vs 12-month-lead-prediction plot for SARIMA model

```
Mean Absolute Error: 0.04603572598026837
Percentage Accuracy: 74.25748693983132 %
PCC : 0.6596829713885668
RMSE : 0.06064966007263505
```

Fig. 4.2. Evaluation metrices of SARIMA model

STL-AR: STL-AR model helps in capturing seasonal and trend components. With autoregressive modelling and advanced time series data decomposition, STL-AR model is a dependable resource for perceptive forecasts and well-informed decision-making.

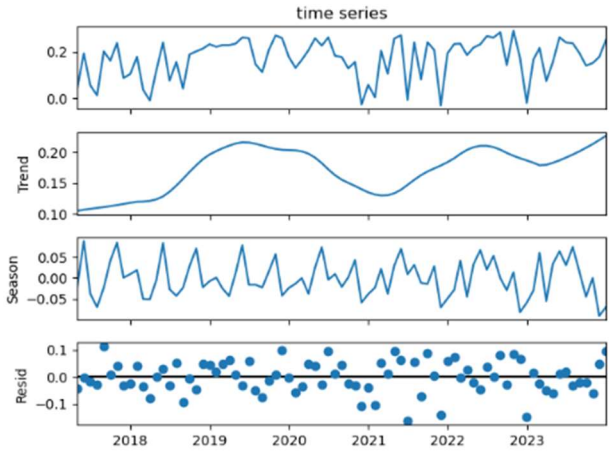


Fig. 4.3. Trend, Seasonal, Residual decomposition using STL

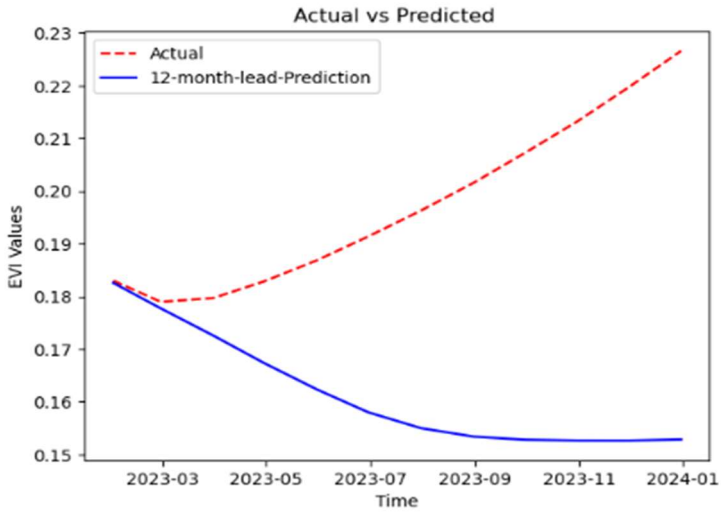


Fig. 4.4. Actual vs 12-month-lead-prediction plot for STL-AR model

```
Mean Absolute Error: 0.0356175908907282
Percentage Accuracy: 81.94542122198358 %
PCC : 0.8079780055584519
RMSE : 0.04338227352534611
```

Fig. 4.5. Evaluation metrices of STL-AR model

DHR: DHR model provides a solid foundation for collecting intricate temporal patterns and producing accurate forecasts. This DHR model is more appropriate way of enhancing forecasting accuracy and supporting well-informed decision-making.

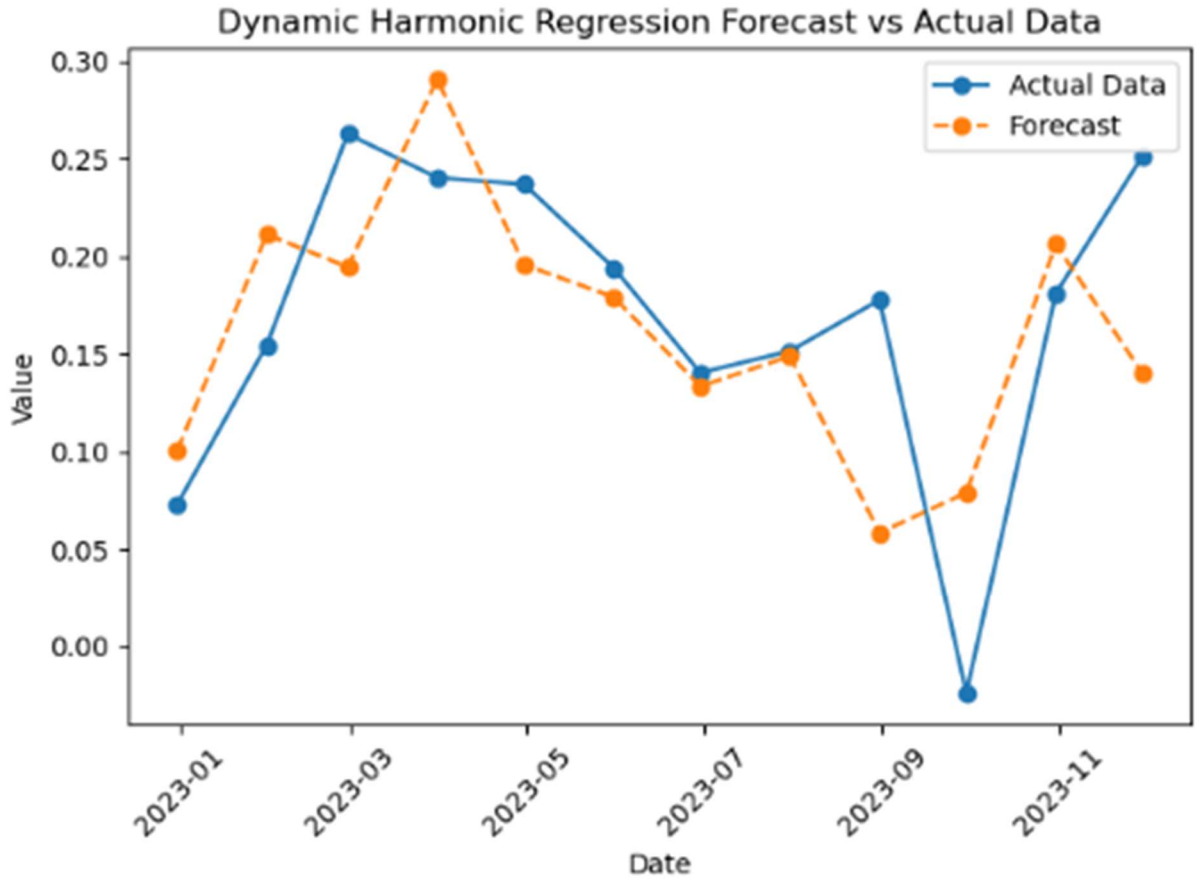


Fig. 4.6. Actual data vs forecast plot for DHR model

```
Mean Absolute Error: 0.052213225583420265
Percentage Accuracy: 69.28824533304696 %
PCC : 0.599310719863862
RMSE : 0.06511104314188855
```

Fig. 4.7. Evaluation metrices of DHR model

TCN: The TCN model shows accurate predictions closely resemble real data. With the help of its creative architecture and deep learning methods, the TCN model is quite good at predicting future observations. This TCN model is a useful tool in your field for making accurate predictions and guiding strategic choices.

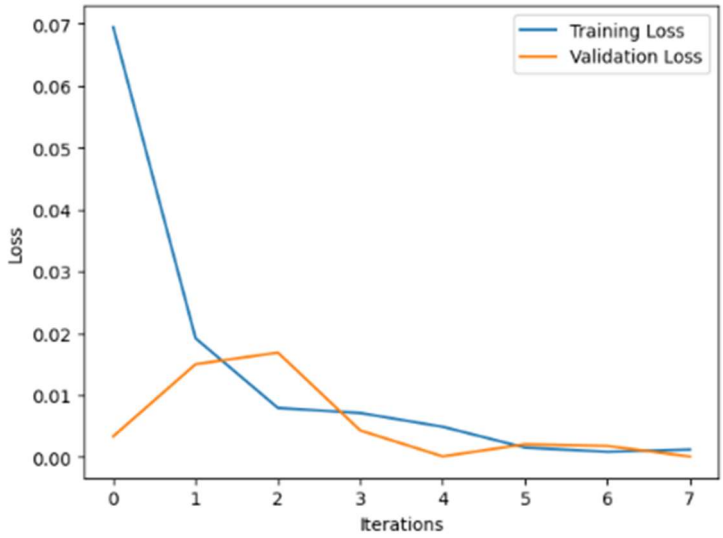


Fig. 4.8. Train loss and validation loss plot for TCN model

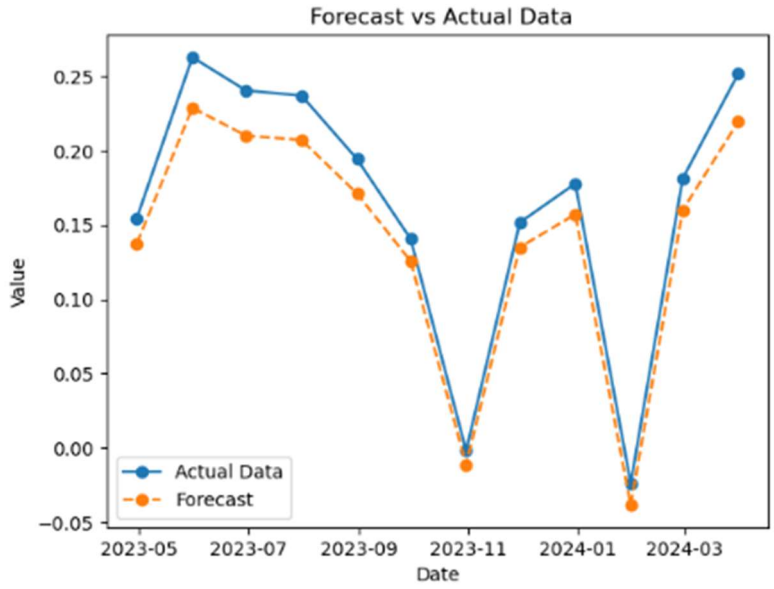


Fig. 4.9. Actual data vs forecast plot for TCN model

```
Mean Absolute Error: 0.021975335034586308
Percentage Accuracy: 86.58138981538963 %
PCC: 0.9989549306575959
RMSE: 0.02327469331269248
```

Fig. 4.10. Evaluation metrices of TCN model

Table 4.1 Comparison of models using performance metrices

Methods	**Paper RMSE**	**Our RMSE**	**Paper PCC**	**Our PCC**
TCN	0.01915	0.0232	0.9960	0.9989
STL-AR	0.0686	0.0447	0.9536	0.8124
SARIMA	0.0554	0.0606	0.9967	0.6596
DHR	0.0598	0.0651	0.9965	0.5993

CHAPTER 5
CONCLUSION AND FUTURE PLANS

The TCN outperformed the remaining three contrasting methods in multiple twelve step prediction, after seeing the results of the other models like STL-AR, SARIMA, and DHR and comparing them using single EVI time series using sentinel 2-hub time series image. The TCN can more effectively extend the change trend of the sequential data and gain more constant prediction results with the real time data.

We used TCN to implement Multiple step satellite image prediction. We aim to expand on this in upcoming times by utilizing LNNs (liquid neural networks). We also aim to demonstrate that LNN produces more accurate findings than TCN.

CHAPTER 6
REFERENCES

1. Advanced road extraction ucin CNN-based U-Net model and satellite imagery
2. Time-series analysis of Sentinel-2 satellite images for sunflower yield estimation.
3. A deforestation detection network using deep learning-based semantic segmentation.
4. Multispectral satellite image segmentation of land cover area using novel naïve bayes in comparison with decision tree.

CHAPTER 7
APPENDIX

BASE PAPER

J. Yan, X. Chen, Y. Chen and D. Liang, "Multistep Prediction of Land Cover From Dense Time Series Remote Sensing Images With Temporal Convolutional Networks," in IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol. 13, pp. 5149-5161, 2020

doi: 10.1109/JSTARS.2020.3020839.

keywords: {Time series analysis; Predictive models; Data models; Remote sensing; Forecasting; Biological system modelling; Prediction algorithms; Dense time series; land use/land cover (LULC); multistep prediction; pixel-level; temporal convolutional networks (TCNs)},

URL:
https://ieeexplore.ieee.org/document/9184116