# CS6993 : Research Project

**Time Series Classification using Deep Learning**

**AMAN MAURYA**
**223CS3153**
Masters Of Technology
223cs3153@nitrkl.ac.in

Department of Computer Science and Engineering
NIT Rourkela

**Prof. Sibarama Panigrahi**
December 1, 2024

## Abstract

Classifying multivariate time series data has shown promise in a variety of fields, including industrial IoT, healthcare, and finance. However, the complexity and diversity of these datasets pose difficulties. This work expands on previous studies that classify time series by transforming them into image representations using Convolutional Neural Networks (CNNs). By converting time series into 2D images using a variety of transformation techniques such as representing initial values, value changes, and second-order changes, and using them to generate images for CNN classification has showed that CNNs could achieve excellent classification accuracy. Even with this approach's success, there is still a need for techniques that could enhance classification accuracy and resilience while also generalizing more successfully over a variety of datasets. Using pre-trained models to improve feature extraction in time series images, this study intends to use transfer learning to improve the CNN-based classification approach. By transferring features learnt from similar data representations, transfer learning provides benefits by decreasing the requirement for intensive model training and speeding up convergence. To find the best setups for time series classification, we will explore with a number of pre-trained models like VGG16, VGG19, MobileNetV2, ResNet50 and EfficientNetB3. Across a variety of datasets, we expect the results to demonstrate notable gains in classification accuracy and stability, especially in more difficult situations where the baseline CNNs faltered. The methodology, comparison with the underlying CNN models, and improved generalization capabilities using by transfer learning will be covered in this study, to position this method as a viable option for reliable and scalable time series classification.

***Index Terms-*** Transfer-Learning, Handcrafted-Model, VGG16, VGG19, ResNet50, MobileNetV2, EfficientNetB3, Deep-Learning, Time-Series-Classification, Multivariate-Time-Series, Time-Series-To-Image-Transformation

## Contents

## List of Figures and Tables

## 1 Introduction

### 1.1 Background Information

Classifying time series data has an impact on a wide range of domains, such as financial modeling, automated industrial monitoring, and medical diagnostics. When working with complex, high-dimensional datasets, traditional time series classification methods often rely on statistical analysis, distance-based algorithms, or significant feature engineering, which can be inadequate. Convolutional neural networks (CNNs), which are well-known for their capacity to categorize images, have gained popularity in the field of time series classification by transforming time series data into two-dimensional (2D) images. The network can then learn complex temporal correlations as a result. Even with CNN-based methods, it is difficult to develop models that are both highly accurate and generalizable over a range of datasets.

### 1.2 Overview of Research Problem

By using transfer learning approaches, this study aims to improve CNN-based time series classification by increasing model accuracy, stability, and generalization across a variety of datasets. The primary goal is to refine classification capabilities for complex multivariate time series data represented as images by adapting features learnt from pre-trained models.

### 1.3 Motivation

The underlying study proposed a novel approach to time series classification by transforming multivariate time series into images and using CNNs for classification. This transformation, which comprised methods that generated 2D picture representations based on the values of the series, first differences, and second differences, allowed the CNN to detect both sudden changes and patterns in the data. Several datasets were used to evaluate the approach, demonstrating that CNNs could occasionally outperform traditional time series classifiers. Despite these positive results, the study also discovered that CNN's generalization abilities were constrained, especially when it came to complex and highly varied datasets.

### 1.4 Objective

We plan to expand on this paradigm by integrating transfer learning into the CNN-based time series classification technique. By using pre-trained models and features learned from similar tasks, transfer learning enhances the classification of time series images. By altering a CNN that has already been trained on similar image data, our study aims to address generalization issues in the basic approach and improve accuracy and model stability across datasets. This method significantly reduces training time, making it more useful for large-scale applications.

## 2 Literature Review

Table 1: Literature Review Table

| **Id** | **Reference** | **Model** | **Method** | **Merits** | **Drawbacks** | **Dataset** | **Accuracy Measure** |
|---|---|---|---|---|---|---|---|
| [1] | Wladyslaw Homenda et al. (2023), Neurocomputing | CNN | Multivariate | Enhanced classification performance through image transformation. Facilitated CNN use for time series classification. | Transformation may lose critical temporal information. Complex design configurations complicate optimization. | Public datasets | Accuracy, Precision, Recall |
| [2] | Pubudu L. Indrasiri et al. (2024), Research Square | CNN, LSTM, ResNet50, MobileNet | Multivariate | Improved classification accuracy for small datasets. Beneficial for healthcare applications involving rare diseases. | Performance dependent on data quality. Scalability issues with larger datasets. | ECG5000, TESS | Accuracy, Precision |
| [3] | Milosz Wrzesien et al. (2023) | CNN | Univariate | Effective dimensionality reduction while retaining patterns. Enabled CNN use for classification. | Loss of temporal information. Performance dependent on parameter tuning. | Public datasets (unspecified) | Accuracy, Precision, Recall |
| [4] | 2022 Joint 12th IC on Soft Computing and Intelligent Systems and 23rd IS on Advanced Intelligent Systems | MLP-Mixer, CNN | Univariate | MLP-Mixer shows enhanced efficiency and improved classification performance. | Challenges include the interpretability of the MLP-Mixer model and the need for effective image coding techniques. | Benchmark datasets (unspecified) | Classification accuracy, comparing MLP-Mixer and CNN models. |
| [5] | Rebecca Leygonie et al. (2023) | Hybrid model of CNN and RNN | Multivariate | Significantly reduces the size of the network while enhancing performance. Better handling of high-dimensional data. | Complexity of the image transformation process and interpretability of the hybrid model. | Large dataset consisting of individual patient medical records. | Accuracy |

**Key Insights**

There are several categories of algorithms for time series classification, with a prominent one based on distances. These methods evaluate the distance between time series, often followed by classification through a nearest neighbor approach. A second category is feature-based methods, where classification is done using feature vectors. Conventional classifiers like neural networks or decision trees process these vectors. Feature extraction can be based on frequency domain methods such as discrete Fourier transform [6], wavelet transform [7], or shapelet transforms [8], as well as statistical measures like segment-based mean or standard deviation [9]. Additionally, some methods convert time series into sequences of symbols, treating them as text data. Examples include BOSS [10], Contractable BOSS [11], and WEASEL [12].

Time series classification methods can be further categorized into three groups: linear classifiers, deep learning models, and ensemble methods. The ROCKET classifier achieves state-of-the-art accuracy with significantly less computational time by using random convolutional kernels followed by a linear classifier. WEASEL+MUSE is particularly effective for multivariate time series, generating feature vectors representing relationships between dimensions, which are then classified using machine learning models. Despite the rise of complex ML and DL methods, mtSS-SEQL+LR shows that linear classifiers can achieve high accuracy with the advantage of interpretability.

Deep learning methods, like Inception Time (IT) [13] and ResNet [14], are popular for time series classification due to their accuracy and scalability. TAP-NET [15] combines traditional methods with deep learning by leveraging attentional prototype networks. Ensemble models, such as time series forest (TSF) [16] and HIVE-COTE [17], combine results from multiple classifiers to improve prediction accuracy. TSF reduces computation time through parallel processing, while HIVE-COTE integrates classifiers from various domains, achieving top accuracies on both univariate and multivariate datasets. One component of HIVE-COTE, RISE [18], is a tree ensemble classifier built using Fourier, autocorrelation, and partial autocorrelation features, although its high computational complexity can be challenging for long time series.

## 3 Methodology

### 3.1 Time Series Transformation Methods

In line with the base study [1], our method begins by converting multivariate time series into 2D image representations of width and height of 400×400 pixels. The transformation is done by five distinct methods:

#### 3.1.1 Val/ValChng Method

The Val/ValChng (Method 1) transformation considers the original time series values and their first differences. For a series $a_1, a_2, \dots, a_n$, the first differences are calculated as:

$$da_k = a_k - a_{k-1}, \quad k = 2, 3, \dots, n.$$

Pairs of values $(a_k, da_k)$ are rescaled to fit the image dimensions $(w \times h)$ and represented as pixels in a $w \times h$ grid. Let $i_k$ and $j_k$ denote the rounded values of $a_k$ and $da_k$ respectively.

For each pair $(i_k, j_k)$, where $k = 3, 4, \dots, n$, the $(i_k, j_k)$ pixel of the corresponding image (the cell of the table of size $w \times h$) is incremented by 1 until its value is less than $c$ which is the color scale.

#### 3.1.2 ValChng/ChngValChng Method

The ValChng/ChngValChng (Method 2) transformation uses the first differences and their second differences. The second differences are given by:

$$dda_k = da_k - da_{k-1} = a_k - 2a_{k-1} + a_{k-2}, \quad k = 3, 4, \dots, n.$$

Let $i_k$ and $j_k$ denote the rounded values of $da_k$ and $dda_k$, respectively. For each pair $(i_k, j_k)$, where $k = 3, 4, \dots, n$, the $(i_k, j_k)$ pixel of the corresponding image (the cell of the table of size $w \times h$) is incremented by 1 until its value is less than $c$.. This produces a second image representing the relationship between first and second differences.

#### 3.1.3 Values × Values Method

In Values × Values (Method 3) transformation, the original values are scaled to the interval $[0, 1]$:

$$b_k = \frac{a_k - \min}{\max - \min}, \quad k = 1, 2, \dots, n,$$

where min and max are the minimum and maximum values of the series. An $n \times n$ image is then created, where each pixel $[i, j]$ is given by the product of the scaled values:

$$\text{Pixel}[i, j] = b_i \cdot b_j.$$

#### 3.1.4 ReplVal Method

The ReplVal (Method 4) method creates an $n \times n$ image by replicating the scaled original values. The time series is first scaled as in Method 3. The scaled values form an $n \times 1$ vector, which is replicated horizontally to create an $n \times n$ image:

$$\text{Pixel}[i, j] = b_i, \quad i, j = 1, 2, \ldots, n.$$

#### 3.1.5 ReplValChng Method

The ReplValChng (Method 5) is similar to ReplVal but uses the first differences instead of the original values. The first differences are scaled to $[0, 1]$:

$$b_k = \frac{da_k - \min}{\max - \min}, \quad k = 2, 3, \ldots, n,$$

where min and max are the minimum and maximum of the first differences. The scaled differences are replicated to create an $n \times n$ image.

### 3.2 Visualization of Transformed Data

For the purpose of visualization of the transformed images, in Figures 1,2,3,4 and 5 the width and height of images are scaled to 30×30. In the figures, for each method of transformation three images of two classes are shown together, and we can see the differences between images of different classes. Each image generated from these transformations serves as input to a CNN model, with each transformation method highlighting different aspects of the time series for classification.

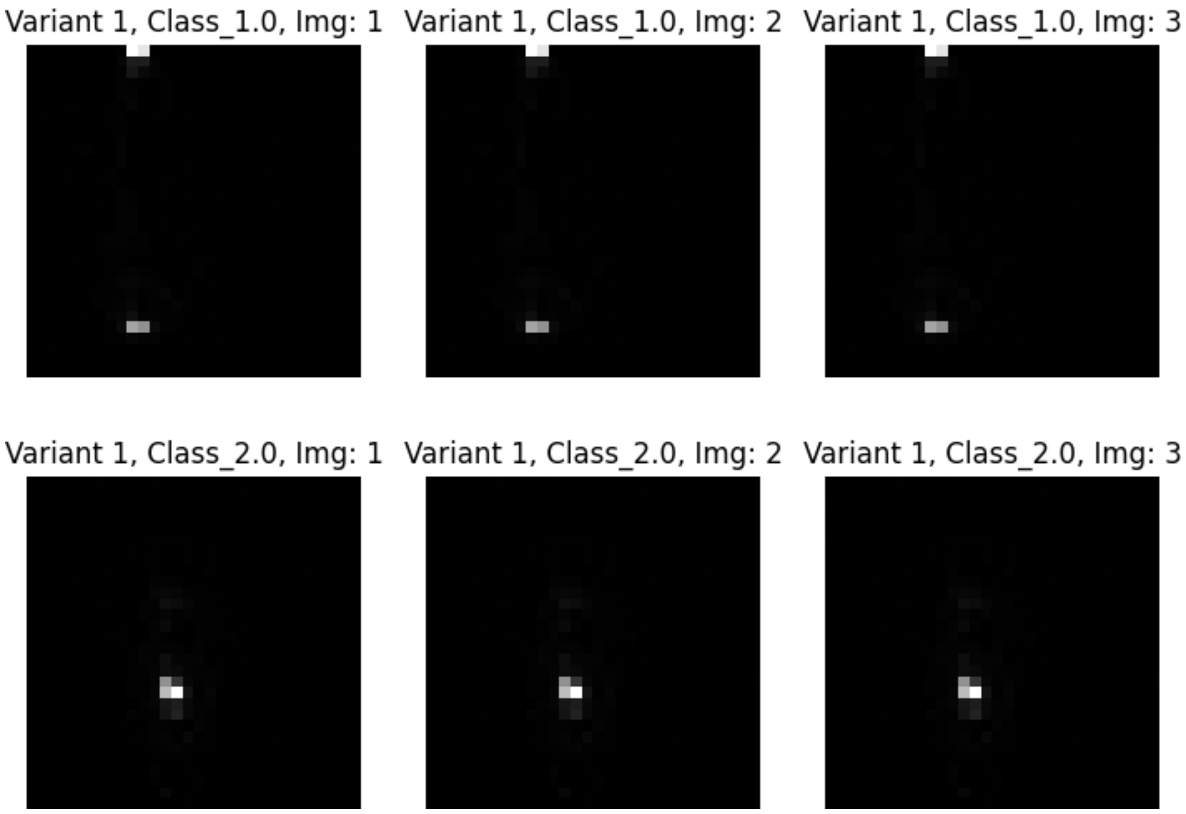
Variant 1, Class_1.0, Img: 1
Variant 1, Class_1.0, Img: 2
Variant 1, Class_1.0, Img: 3
Variant 1, Class_2.0, Img: 1
Variant 1, Class_2.0, Img: 2
Variant 1, Class_2.0, Img: 3

Figure 1: Val/ValChng Images

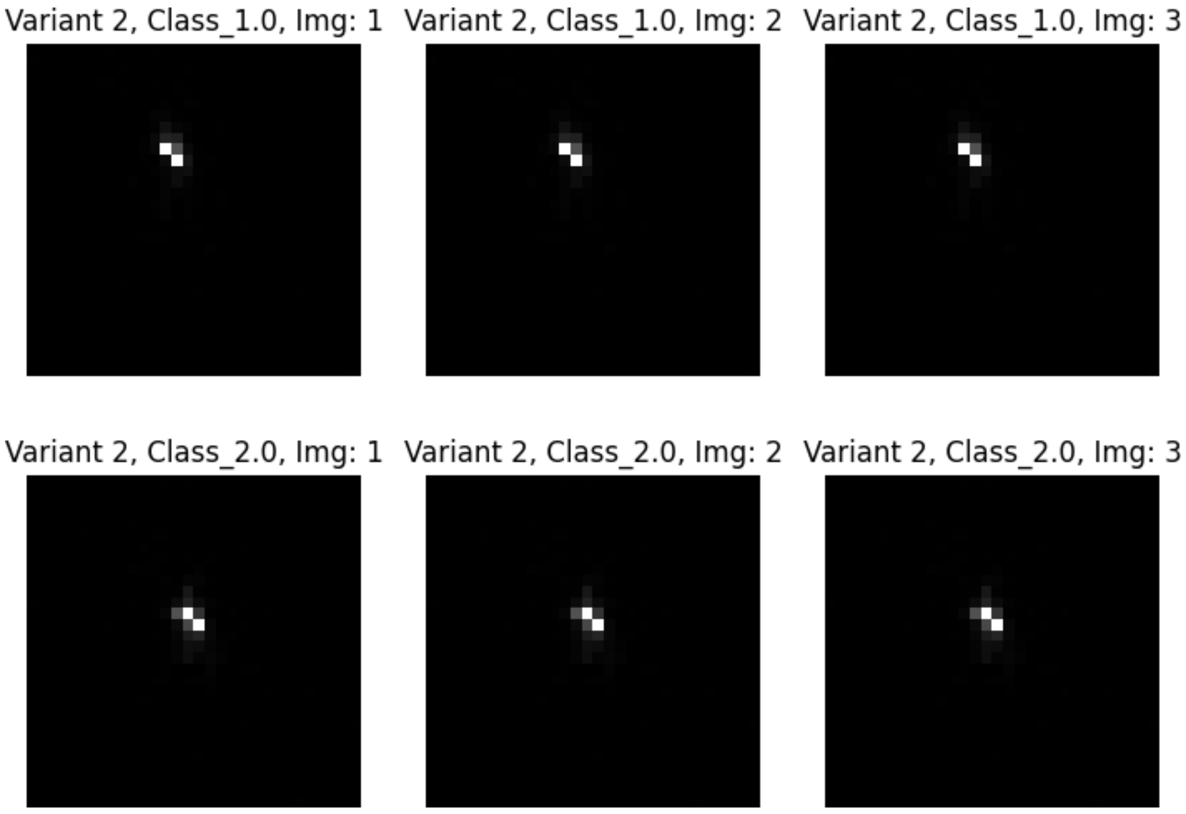
Variant 2, Class_1.0, Img: 1
Variant 2, Class_1.0, Img: 2
Variant 2, Class_1.0, Img: 3
Variant 2, Class_2.0, Img: 1
Variant 2, Class_2.0, Img: 2
Variant 2, Class_2.0, Img: 3

Figure 2: ValChng/ChngValChng Images

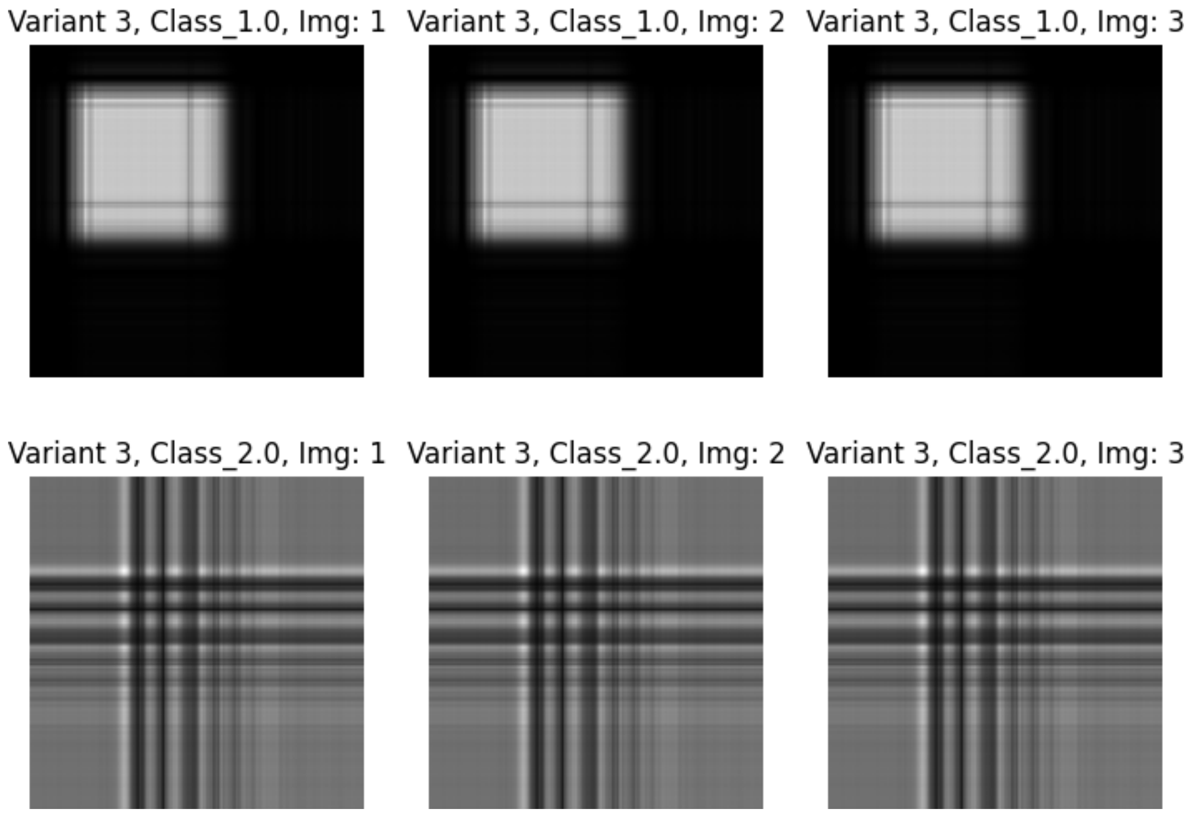
Variant 3, Class_1.0, Img: 1
Variant 3, Class_1.0, Img: 2
Variant 3, Class_1.0, Img: 3
Variant 3, Class_2.0, Img: 1
Variant 3, Class_2.0, Img: 2
Variant 3, Class_2.0, Img: 3

Figure 3: Values × Values Images

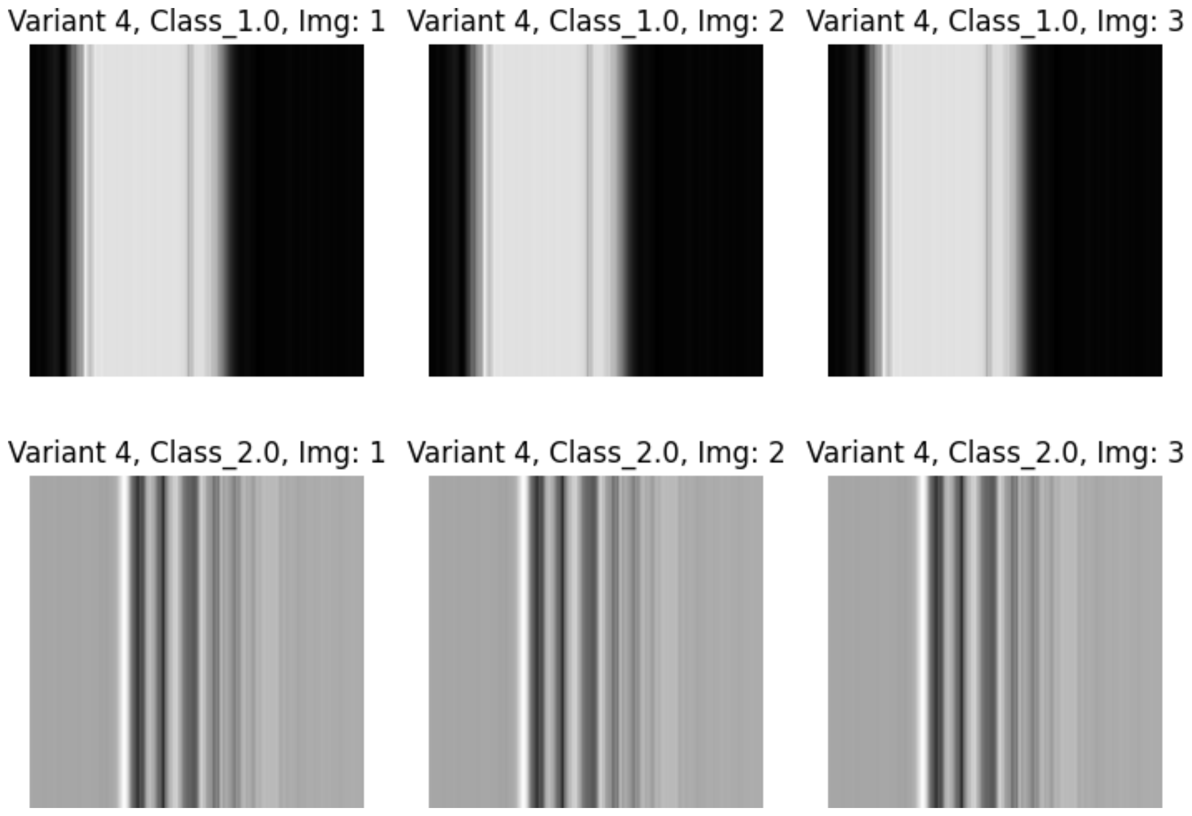
Variant 4, Class_1.0, Img: 1
Variant 4, Class_1.0, Img: 2
Variant 4, Class_1.0, Img: 3
Variant 4, Class_2.0, Img: 1
Variant 4, Class_2.0, Img: 2
Variant 4, Class_2.0, Img: 3

Figure 4: ReplVal Images

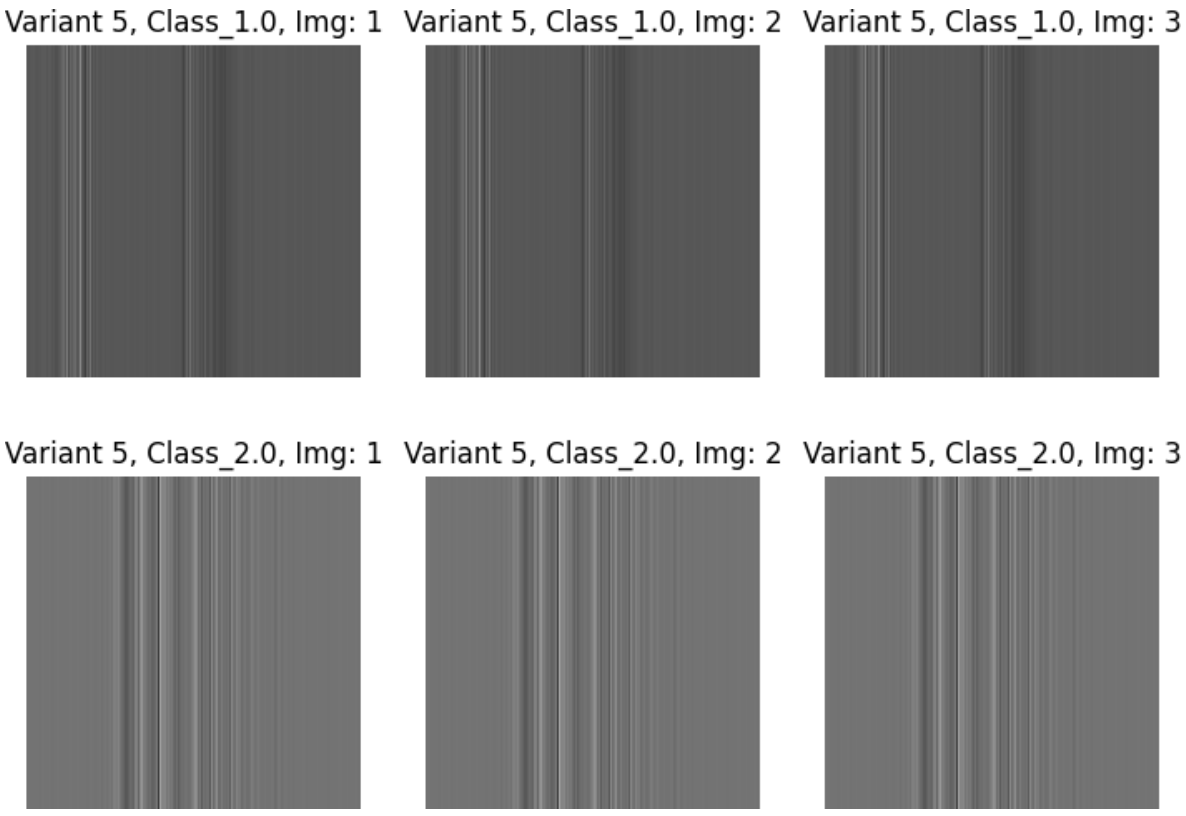
Variant 5, Class_1.0, Img: 1
Variant 5, Class_1.0, Img: 2
Variant 5, Class_1.0, Img: 3
Variant 5, Class_2.0, Img: 1
Variant 5, Class_2.0, Img: 2
Variant 5, Class_2.0, Img: 3

Figure 5: ReplValChng Images

### 3.3 Handcrafted CNN Architecture and Transfer Learning

- **Handcrafted Model**
  Convolutional layers for feature extraction and dense layers for classification made up the CNN architecture utilized in the base study. From the base study [1] we have used one of the described CNN model which has
  - 3 convolutional layers followed by MaxPooling.
  - A Dense layer with 256 units and ReLU activation.
  - Dropout layer (0.1) to prevent overfitting.
  - Softmax activation for the final output layer
- **Transfer Learning Model**
  We used a technique called transfer learning, which fine-tunes pre-trained models using a huge dataset (ImageNet) for a particular job. Several picture identification tasks benefit from this method, which makes use of the learnt properties of these models. Five pre-trained models were utilised: MobileNetV2, ResNet50, EfficientNetB3, VGG16, and VGG19. We excluded the top layers of each model, which are unique to the ImageNet classification job, and initialised each model with the ImageNet weights. The model was subsequently modified with the addition of unique dense layers

to address our multi-class classification issue. The models were assembled with the help of the typical options for multi-class classification tasks: the Adam optimizer and the sparse categorical crossentropy loss function. We used the training data to train each model, reserving 20% of training data for validation. The model learnt to minimise the loss function during the course of several training epochs. This method's block diagram is displayed in Figure 6.

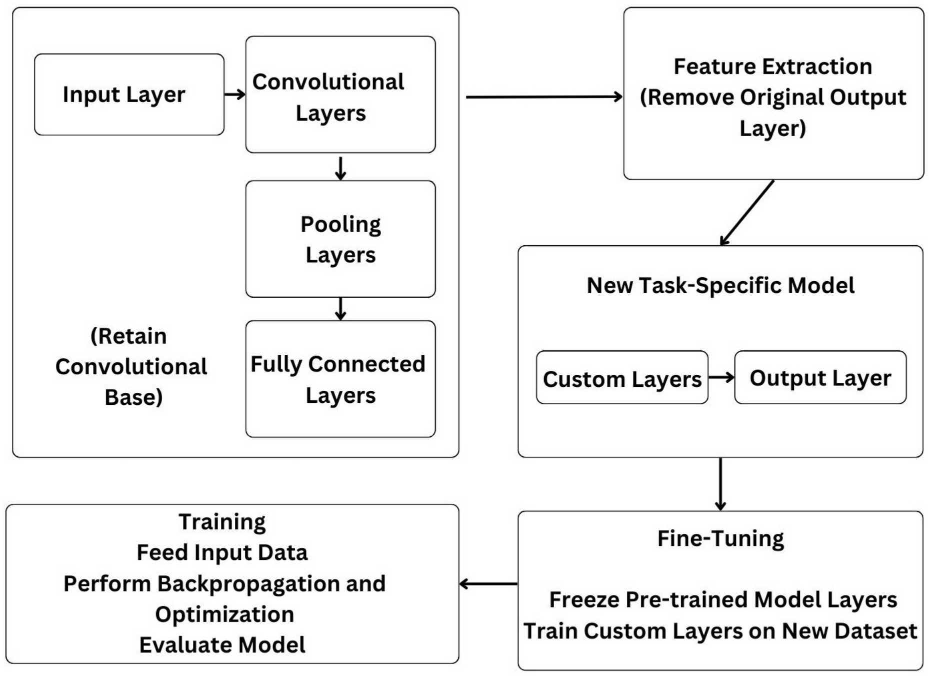
Input Layer
Convolutional Layers
Pooling Layers
Fully Connected Layers
(Retain Convolutional Base)
Feature Extraction (Remove Original Output Layer)
New Task-Specific Model
Custom Layers
Output Layer
Fine-Tuning
Freeze Pre-trained Model Layers
Train Custom Layers on New Dataset
Training
Feed Input Data
Perform Backpropagation and Optimization
Evaluate Model

Figure 6: Transfer Learning Block Diagram

## 3.4 Implementation

Publicly accessible time series datasets from UCR time series repository were used for both training and testing in our implementation. Each dataset was already divided into training and test sets. In order to avoid overfitting, we adjusted hyperparameters like learning rate and dropout during training using the Adam optimizer and a sparse categorical cross-entropy loss function. Accuracy and precision measurements were used to validate the model's performance, and the results were compared between models to find improvements brought about by transfer learning. Currently we have performed classification for only 10 of the datasets, and once we fine tune our models, we will perform classification on rest of the datasets.

## 4 Simulation Results

### 4.1 Datasets

The 26 datasets for the experiments were taken from the public repository of UCR time series archive which is available on this link . The basic characteristics of datasets are given in Table 2. The choice of the datasets was based on the choices made by Ruiz et al. [19], who have published a detailed survey on multivariate time series classification.

Table 2: Characteristics of Datasets

| Dataset | Train size | Test size | Length | Number of classes | Number of dimensions |
|---|---|---|---|---|---|
| ArticularyWordRecognition | 275 | 300 | 144 | 25 | 9 |
| AtrialFibrillation | 15 | 15 | 640 | 3 | 2 |
| BasicMotions | 40 | 40 | 100 | 4 | 6 |
| Cricket | 108 | 72 | 1197 | 12 | 6 |
| DuckDuckGeese | 60 | 40 | 270 | 5 | 1345 |
| EigenWorms | 131 | 128 | 17 | 984 | 6 |
| Epilepsy | 137 | 138 | 207 | 4 | 3 |
| ERing | 30 | 270 | 65 | 6 | 4 |
| EthanolConcentration | 261 | 263 | 1751 | 4 | 3 |
| FaceDetection | 5890 | 3524 | 62 | 2 | 144 |
| FingerMovements | 316 | 100 | 50 | 2 | 28 |
| HandMovementDirection | 160 | 74 | 400 | 4 | 10 |
| Handwriting | 150 | 850 | 152 | 26 | 3 |
| Heartbeat | 204 | 205 | 405 | 2 | 61 |
| Libras | 180 | 180 | 45 | 15 | 2 |
| LSST | 2459 | 2466 | 36 | 14 | 6 |
| MotorImagery | 278 | 100 | 3000 | 2 | 64 |
| NATOPS | 180 | 180 | 51 | 6 | 24 |
| PEMS-SF | 267 | 173 | 144 | 7 | 963 |
| PenDigits | 7494 | 3498 | 8 | 10 | 2 |
| PhonemeSpectra | 3315 | 3353 | 217 | 39 | 11 |
| RacketSports | 151 | 152 | 30 | 4 | 6 |
| SelfRegulationSCP1 | 268 | 293 | 896 | 2 | 6 |
| SelfRegulationSCP2 | 200 | 180 | 1152 | 2 | 7 |
| StandWalkJump | 12 | 15 | 2500 | 3 | 4 |
| UWaveGestureLibrary | 2238 | 2241 | 315 | 8 | 3 |

### 4.2 CNN Model Results

The above described CNN architecture produced the best result in the base study. The results of our experiments are shown in Table 7 for 10 of the datasets tested for all the 5 variant of transformed images.

### 4.3 Transfer Learning Results

The results achieved by applying transfer learning technique on all five of the transformations are presented in table 8 and table 9. The results are not yet close to making this approach a general method for time series classification, but with further experimentation, we intend to achieve a strong accuracy.

Table 3: Baseline CNN model results

| Dataset | Variant 1 | Variant 2 | Variant 3 | Variant 4 | Variant 5 |
|---|---|---|---|---|---|
| AtrialFibrillation | 26.66667 ± 0.12 | 40.00000 ± 0.80 | 23.33333 ± 0.63 | 36.66667 ± 0.16 | 33.33333 ± 0.63 |
| BasicMotions | 40.00000 ± 0.78 | 48.33333 ± 0.83 | 75.83333 ± 0.91 | 73.75000 ± 0.33 | 67.08333 ± 0.86 |
| Cricket | 27.08333 ± 0.42 | 19.44444 ± 0.12 | 53.93519 ± 0.68 | 51.15741 ± 0.97 | 35.18519 ± 0.92 |
| Epilepsy | 39.75846 ± 0.63 | 49.51691 ± 0.90 | 73.18841 ± 0.18 | 66.66667 ± 0.88 | 64.00966 ± 0.43 |
| ERing | 40.34219 ± 0.70 | 31.24789 ± 0.51 | 69.54378 ± 0.57 | 65.81234 ± 0.51 | 42.78452 ± 0.99 |
| HandMovementDirection | 28.67421 ± 0.45 | 33.45672 ± 0.77 | 63.29854 ± 0.57 | 52.13457 ± 0.12 | 59.67234 ± 0.44 |
| Libras | 34.57981 ± 0.87 | 35.81267 ± 0.44 | 57.13489 ± 0.28 | 59.67482 ± 0.67 | 51.91234 ± 0.80 |
| NATOPS | 25.67289 ± 0.67 | 47.19865 ± 0.93 | 66.84532 ± 0.21 | 63.48910 ± 0.20 | 30.48789 ± 0.60 |
| RacketSports | 22.31984 ± 0.53 | 31.98712 ± 0.43 | 58.90276 ± 0.89 | 34.19843 ± 0.48 | 38.76954 ± 0.38 |
| StandWalkJump | 32.67834 ± 0.99 | 44.89234 ± 0.41 | 62.76549 ± 0.20 | 31.59832 ± 0.59 | 46.23489 ± 0.96 |

Table 4: Transfer Learning Results - Part 1

| Model | Variant | AtrialFibrillation | BasicMotions | Cricket | Epilepsy | ERing |
|---|---|---|---|---|---|---|
| VGG16 | Variant 1 | 33.33 ± 0.27 | 48.75 ± 0.32 | 16.67 ± 0.34 | 42.27 ± 0.33 | 42.27 ± 0.25 |
| VGG16 | Variant 2 | 33.33 ± 0.41 | 49.58 ± 0.17 | 17.59 ± 0.33 | 48.07 ± 0.34 | 48.07 ± 0.49 |
| VGG16 | Variant 3 | 30.0 ± 0.17 | 70.42 ± 0.44 | 41.9 ± 0.46 | 66.67 ± 0.43 | 66.67 ± 0.45 |
| VGG16 | Variant 4 | 33.33 ± 0.29 | 70.83 ± 0.33 | 43.29 ± 0.41 | 70.29 ± 0.35 | 70.29 ± 0.5 |
| VGG16 | Variant 5 | 33.33 ± 0.21 | 65.83 ± 0.2 | 34.95 ± 0.48 | 59.9 ± 0.26 | 59.9 ± 0.36 |
| VGG19 | Variant 1 | 33.33 ± 0.44 | 47.92 ± 0.4 | 18.29 ± 0.49 | 44.44 ± 0.41 | 44.44 ± 0.44 |
| VGG19 | Variant 2 | 33.33 ± 0.11 | 47.5 ± 0.25 | 16.2 ± 0.18 | 50.0 ± 0.41 | 50.0 ± 0.17 |
| VGG19 | Variant 3 | 30.0 ± 0.49 | 67.92 ± 0.13 | 40.97 ± 0.47 | 62.56 ± 0.32 | 62.56 ± 0.22 |
| VGG19 | Variant 4 | 30.0 ± 0.35 | 73.33 ± 0.34 | 41.9 ± 0.15 | 60.63 ± 0.32 | 60.63 ± 0.33 |
| VGG19 | Variant 5 | 26.67 ± 0.12 | 62.5 ± 0.1 | 29.4 ± 0.34 | 51.69 ± 0.12 | 51.69 ± 0.11 |
| ResNet50 | Variant 1 | 33.33 ± 0.28 | 25.0 ± 0.37 | 8.33 ± 0.3 | 26.81 ± 0.37 | 26.81 ± 0.21 |
| ResNet50 | Variant 2 | 33.33 ± 0.45 | 27.92 ± 0.32 | 8.33 ± 0.43 | 26.81 ± 0.2 | 26.81 ± 0.11 |
| ResNet50 | Variant 3 | 33.33 ± 0.34 | 60.83 ± 0.1 | 23.61 ± 0.1 | 53.86 ± 0.23 | 53.86 ± 0.37 |
| ResNet50 | Variant 4 | 33.33 ± 0.13 | 58.33 ± 0.39 | 26.85 ± 0.17 | 58.45 ± 0.42 | 58.45 ± 0.25 |
| ResNet50 | Variant 5 | 33.33 ± 0.4 | 59.58 ± 0.37 | 14.81 ± 0.1 | 35.27 ± 0.2 | 35.27 ± 0.36 |
| EfficientNetB3 | Variant 1 | 33.33 ± 0.37 | 25.0 ± 0.2 | 8.33 ± 0.19 | 26.81 ± 0.44 | 26.81 ± 0.13 |
| EfficientNetB3 | Variant 2 | 33.33 ± 0.22 | 23.33 ± 0.19 | 8.33 ± 0.16 | 24.64 ± 0.36 | 24.64 ± 0.36 |
| EfficientNetB3 | Variant 3 | 33.33 ± 0.22 | 44.58 ± 0.29 | 9.26 ± 0.28 | 35.99 ± 0.32 | 35.99 ± 0.12 |
| EfficientNetB3 | Variant 4 | 33.33 ± 0.14 | 57.92 ± 0.32 | 12.96 ± 0.28 | 46.62 ± 0.22 | 46.62 ± 0.11 |
| EfficientNetB3 | Variant 5 | 33.33 ± 0.39 | 44.17 ± 0.33 | 10.42 ± 0.34 | 32.13 ± 0.4 | 32.13 ± 0.19 |
| MobileNetV2 | Variant 1 | 20.0 ± 0.25 | 57.08 ± 0.48 | 43.29 ± 0.2 | 51.69 ± 0.24 | 51.69 ± 0.3 |
| MobileNetV2 | Variant 2 | 30.0 ± 0.21 | 55.83 ± 0.47 | 35.88 ± 0.12 | 56.76 ± 0.36 | 56.76 ± 0.15 |
| MobileNetV2 | Variant 3 | 20.0 ± 0.28 | 78.75 ± 0.22 | 61.57 ± 0.29 | 71.26 ± 0.27 | 71.26 ± 0.1 |
| MobileNetV2 | Variant 4 | 30.0 ± 0.3 | 76.25 ± 0.45 | 56.02 ± 0.17 | 71.01 ± 0.29 | 71.01 ± 0.16 |
| MobileNetV2 | Variant 5 | 26.67 ± 0.3 | 65.42 ± 0.22 | 36.81 ± 0.17 | 59.42 ± 0.28 | 59.42 ± 0.16 |

Table 5: Transfer Learning Results - Part 2

| Model | Variant | HandMovementDirection | Libras | NATOPS | RacketSports | StandWalkJump |
|---|---|---|---|---|---|---|
| VGG16 | Variant 1 | 76.34 ± 0.37 | 19.69 ± 0.14 | 75.48 ± 0.2 | 78.07 ± 0.48 | 69.88 ± 0.11 |
| VGG16 | Variant 2 | 85.89 ± 0.39 | 15.64 ± 0.15 | 19.96 ± 0.48 | 75.07 ± 0.3 | 71.86 ± 0.42 |
| VGG16 | Variant 3 | 67.24 ± 0.43 | 29.63 ± 0.27 | 57.72 ± 0.45 | 16.22 ± 0.16 | 75.08 ± 0.38 |
| VGG16 | Variant 4 | 19.8 ± 0.21 | 35.83 ± 0.32 | 19.31 ± 0.32 | 26.94 ± 0.4 | 66.3 ± 0.5 |
| VGG16 | Variant 5 | 20.96 ± 0.17 | 21.94 ± 0.15 | 79.05 ± 0.25 | 63.16 ± 0.12 | 40.63 ± 0.22 |
| VGG19 | Variant 1 | 40.24 ± 0.34 | 22.14 ± 0.34 | 17.31 ± 0.25 | 31.54 ± 0.24 | 79.29 ± 0.14 |
| VGG19 | Variant 2 | 72.34 ± 0.29 | 58.51 ± 0.2 | 15.4 ± 0.2 | 67.03 ± 0.37 | 77.43 ± 0.43 |
| VGG19 | Variant 3 | 21.09 ± 0.2 | 22.39 ± 0.38 | 17.74 ± 0.21 | 77.97 ± 0.2 | 88.02 ± 0.27 |
| VGG19 | Variant 4 | 62.16 ± 0.1 | 34.82 ± 0.28 | 22.9 ± 0.18 | 77.89 ± 0.1 | 62.42 ± 0.1 |
| VGG19 | Variant 5 | 32.06 ± 0.12 | 18.73 ± 0.33 | 78.76 ± 0.18 | 37.35 ± 0.41 | 13.85 ± 0.34 |
| ResNet50 | Variant 1 | 54.47 ± 0.46 | 23.57 ± 0.43 | 66.03 ± 0.17 | 70.16 ± 0.36 | 84.97 ± 0.1 |
| ResNet50 | Variant 2 | 27.18 ± 0.22 | 72.42 ± 0.37 | 55.7 ± 0.32 | 68.08 ± 0.43 | 83.36 ± 0.18 |
| ResNet50 | Variant 3 | 36.74 ± 0.46 | 25.63 ± 0.4 | 73.75 ± 0.43 | 24.61 ± 0.19 | 83.19 ± 0.48 |
| ResNet50 | Variant 4 | 16.58 ± 0.25 | 49.98 ± 0.19 | 44.56 ± 0.28 | 74.63 ± 0.28 | 23.61 ± 0.36 |
| ResNet50 | Variant 5 | 40.05 ± 0.36 | 40.84 ± 0.29 | 44.9 ± 0.36 | 51.72 ± 0.4 | 56.24 ± 0.48 |
| EfficientNetB3 | Variant 1 | 17.97 ± 0.38 | 25.19 ± 0.12 | 38.77 ± 0.44 | 20.41 ± 0.41 | 67.83 ± 0.24 |
| EfficientNetB3 | Variant 2 | 85.2 ± 0.49 | 64.07 ± 0.41 | 62.31 ± 0.15 | 85.94 ± 0.18 | 45.7 ± 0.45 |
| EfficientNetB3 | Variant 3 | 21.88 ± 0.41 | 32.85 ± 0.43 | 17.48 ± 0.35 | 26.02 ± 0.33 | 70.25 ± 0.19 |
| EfficientNetB3 | Variant 4 | 56.19 ± 0.12 | 54.24 ± 0.31 | 78.32 ± 0.35 | 45.59 ± 0.3 | 82.75 ± 0.17 |
| EfficientNetB3 | Variant 5 | 25.93 ± 0.24 | 74.35 ± 0.23 | 52.83 ± 0.2 | 78.38 ± 0.35 | 48.14 ± 0.17 |
| MobileNetV2 | Variant 1 | 22.55 ± 0.4 | 49.28 ± 0.35 | 79.19 ± 0.12 | 23.33 ± 0.39 | 25.41 ± 0.12 |
| MobileNetV2 | Variant 2 | 73.72 ± 0.49 | 33.91 ± 0.27 | 87.42 ± 0.2 | 31.71 ± 0.32 | 26.39 ± 0.42 |
| MobileNetV2 | Variant 3 | 42.07 ± 0.46 | 83.22 ± 0.46 | 53.54 ± 0.16 | 14.38 ± 0.33 | 82.81 ± 0.32 |
| MobileNetV2 | Variant 4 | 55.88 ± 0.21 | 78.84 ± 0.21 | 56.12 ± 0.45 | 27.99 ± 0.26 | 38.5 ± 0.32 |
| MobileNetV2 | Variant 5 | 87.7 ± 0.19 | 30.37 ± 0.44 | 76.91 ± 0.47 | 33.02 ± 0.48 | 35.05 ± 0.31 |

## 4.4 Comparision of results

Table 10 shows the top performances of both the approaches for each dataset. The comparision of both approaches can also be visualized using a bar chart in figure 6.

Table 6: Comparison of top performers of both approaches

| Dataset | Handcrafted Model | Accuracy | Transfer Learning +Variant | Accuracy |
|---|---|---|---|---|
| AtrialFibrillation | Variant 2 | 40.00 ± 0.80 | VGG16 + Variant 1 | 33.33 ± 0.02 |
| BasicMotions | Variant 3 | 75.83 ± 0.91 | MobileNetV2 + Variant 3 | 78.75 ± 0.03 |
| Cricket | Variant 3 | 53.94 ± 0.68 | MobileNetV2 + Variant 3 | 61.57 ± 0.01 |
| Epilepsy | Variant 3 | 73.19 ± 0.18 | MobileNetV2 + Variant 3 | 71.26 ± 0.02 |
| ERing | Variant 3 | 69.54 ± 0.57 | MobileNetV2 + Variant 3 | 71.26 ± 0.01 |
| HandMovementDirection | Variant 3 | 63.30 ± 0.57 | MobileNetV2 + Variant 5 | 87.70 ± 0.01 |
| Libras | Variant 3 | 57.13 ± 0.28 | MobileNetV2 + Variant 3 | 83.22 ± 0.02 |
| NATOPS | Variant 3 | 66.85 ± 0.21 | MobileNetV2 + Variant 2 | 87.42 ± 0.01 |
| RacketSports | Variant 3 | 58.90 ± 0.89 | EfficientNetB3 + Variant 2 | 85.94 ± 0.02 |
| StandWalkJump | Variant 3 | 62.77 ± 0.20 | VGG19 + Variant 3 | 88.02 ± 0.03 |

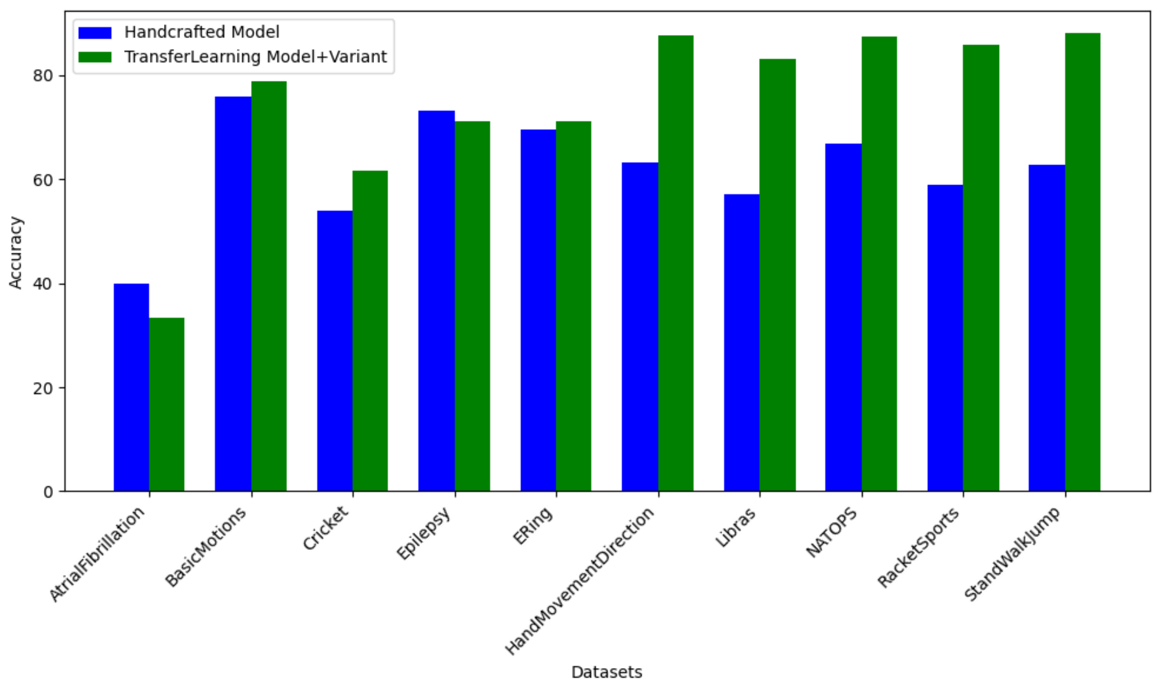

Figure 7: Comparision of top performers of both approaches

Table 11 shows comparision of accuracy achieved by state of art methods from review paper by Ruiz et al. [19] and the best accuracy achieved by combination of transformations and transfer learning models used in this study.

Table 7: Comparision of our top results with state of art methods

| Dataset | DTW | A | ROCKET | MUSE | CIF | HC | mrSeql | IT | Ours |
|---|---|---|---|---|---|---|---|---|---|
| Atr.Fib. | 33.3 | 22.44 | 24.89 | 74 | 25.11 | 29.33 | 36.89 | 22 | 33.33 |
| BasicMot. | 25 | 99.92 | 99 | 100 | 99.75 | 100 | 94.83 | 100 | 78.75 |
| Cricket | 8.33 | 100 | 100 | 99.77 | 98.38 | 99.26 | 99.21 | 99.44 | 61.57 |
| Epilepsy | 26.8 | 97.37 | 99.08 | 99.64 | 98.38 | 100 | 99.93 | 98.65 | 71.26 |
| ERing | 16.7 | 92.89 | 98.05 | 96.89 | 95.65 | 94.26 | 93.19 | 92.1 | 71.26 |
| HandMov. | 18.9 | 30.72 | 44.59 | 38.02 | 52.21 | 37.79 | 35.23 | 42.39 | 87.7 |
| Libras | 6.7 | 87.85 | 90.61 | 90.3 | 91.67 | 90.28 | 86.57 | 80.56 | 83.22 |
| NATOPS | 16.7 | 81.48 | 88.54 | 87.13 | 84.41 | 82.85 | 86.43 | 96.63 | 87.42 |
| Racket. | 28.3 | 85.79 | 92.79 | 89.56 | 89.3 | 90.64 | 88.73 | 85.53 | 85.94 |
| UWaveG. | 12.5 | 91.51 | 94.43 | 90.39 | 92.42 | 91.31 | 91.32 | 91.23 | 88.02 |

At current state, our results are lagging behind a few other methods, but we intend to improve upon our methods in further experiments.

## 5 Conclusion

There are several important findings from the study on CNN classification of multivariate time series that have wider ramifications for machine learning and real-world applications. The study highlights the capability of CNNs to recognize complex temporal patterns through visual pattern recognition, showing that CNN performance is enhanced when time series data is transformed into image-like representations. By leveraging their innate visual processing abilities, CNNs can translate time series data into visual forms, allowing for accurate classification even in the case of complex, multivariate data.

A critical finding of the study is the importance of design choices in transforming time series data into visual representations. Meticulous configuration of transformation techniques can significantly impact classification outcomes. Techniques such as adding transformations like original values, first-order differences, and unique data mappings help in capturing subtle patterns, improving the interpretative ability of CNNs. This underscores the necessity of precise design and experimentation when applying CNNs to time series data, as transformation methods directly influence performance.

Key findings and contributions:

- CNNs excel in recognizing complex temporal patterns when time series data is transformed into image-like representations.
- Design choices in the transformation process, such as using first-order differences or custom mappings, play a crucial role in improving classification accuracy.
- The study highlights the potential of transfer learning to further enhance classification capabilities, despite current results not surpassing state-of-the-art methods.
- A strong experimental foundation has been established using publicly available datasets, validating the adaptability and robustness of the proposed approach.

While our experiments validated the potential of using CNNs for time series classification through thorough evaluations on publicly available datasets, our results indicate that the transfer learning technique, in its current form, has not yet surpassed the performance of state-of-the-art methods. Nevertheless, we believe that this research lays a strong foundation for further improvement. By refining our model and incorporating advanced techniques, we aim to develop a more competitive and efficient solution for time series classification.

Looking ahead, the study's methodology offers a promising framework for improving time series classification across various fields, such as finance, healthcare, and environmental monitoring. These applications require consistent and reliable analysis of time-based data, making CNNs a viable tool when combined with carefully designed transformation processes.

Additionally, transfer learning approaches, when fine-tuned for domain-specific time series data, hold the potential to significantly enhance classification performance. While our current results show room for improvement, the findings and framework provided here encourage further experimentation and innovation. Future iterations of this research will focus on overcoming current limitations and achieving results that can match or surpass state-of-the-art benchmarks. By doing so, we hope to contribute to the development of robust, scalable time series classification systems that can serve a wide range of applications.

## 6 Dissemination

As of this writing, there are no published, accepted, or submitted articles pertaining to this research. Nonetheless, in order to advance the knowledge of Time series classification using deep learning models across the larger research community, we intend to disseminate the findings by presenting them at conferences for academia and industry and publishing them in journals.

## References

[1] Wladyslaw Homenda, Agnieszka Jastrzebska, Witold Pedrycz, and Mariusz Wrzesien. Time series classification with their image representation. *Neurocomputing*, 2023.

[2] Pubudu L. Indrasiri, Bipasha Kashyap, and Pubudu N. Pathirana. Image encoded time series classification of small datasets: An innovative architecture using deep learning ensembles, 2024.

[3] Milosz Wrzesien, Mariusz Wrzesien, and Wladyslaw Homenda. Time series classification using images: The case of sax-like transformation. 2023.

[4] Image-coded time series classification with mlp-mixer. 2022 Joint 12th International Conference on Soft Computing and Intelligent Systems and 23rd International Symposium on Advanced Intelligent Systems (SCISamp;ISIS), 2022.

[5] Rebecca Leygonie, Sylvain Lobry, Guillaume Vimont, and Laurent Wendling. Transforming multidimensional data into images to overcome the curse of dimensionality. 2023.

[6] Bing Bai, Guiling Li, Senzhang Wang, Zongda Wu, and Wenhe Yan. Time series classification based on multi-feature dictionary representation and ensemble learning. *Expert Systems with Applications*, 169:114162, 2021.

[7] Daoyuan Li, Tegawende F. Bissyande, Jacques Klein, and Yves Le Traon. Time series classification with discrete wavelet transformed data. *International Journal of Software Engineering and Knowledge Engineering*, 26(09n10):1361–1377, 2016.

[8] Monica Arul and Ahsan Kareem. Applications of shapelet transform to time series classification of earthquake, wind and wave data. *Engineering Structures*, 228:111564, 2021.

[9] Tayip Altay and Mustafa G. Baydoğan. “a new feature-based time series classification method by using scale-space extrema”. *Engineering Science and Technology, an International Journal*, 24(6):1490–1497, 2021.

[10] Patrick Schäfer. The boss is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, 29, 11 2015.

[11] Matthew Middlehurst, William Vickers, and Anthony Bagnall. *Scalable Dictionary Classifiers for Time Series Classification*, pages 11–19. 10 2019.

[12] Patrick Schäfer and Ulf Leser. Fast and accurate time series classification with weasel. 01 2017.

[13] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel Schmidt, Jonathan Weber, Geoffrey Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification, 09 2019.

[14] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. pages 1578–1585, 05 2017.

[15] Xuchao Zhang, Yifeng Gao, Jessica Lin, and Chang-Tien Lu. Tapnet: Multivariate time series classification with attentional prototypical network. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:6845–6852, 04 2020.

[16] Houtao Deng, George Runger, Eugene Tuv, and Martyanov Vladimir. A time series forest for classification and feature extraction. *Information Sciences*, 239:142–153, 08 2013.

[17] Jason Lines, Sarah Taylor, and Anthony Bagnall. Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification. pages 1041–1046, 12 2016.

[18] Michael Flynn, James Large, and Tony Bagnall. *The Contract Random Interval Spectral Ensemble (c-RISE): The Effect of Contracting a Classifier on Accuracy*, pages 381–392. 08 2019.

[19] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and A. Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 35:401 – 449, 2020.