Unified Framework for Time Series Classification

Aayush Sureshchander
Department of Electrical and Computer Engineering
Virginia Polytechnic and State University
Blacksburg, VA 24060
aayush19@vt.edu

Abstract

Time series data is a very vast and easily available data source that researchers are still trying to find optimal classification and prediction algorithms for. Similar to image data, it is natural to want to use convolution networks and deep neural networks towards this task, however this has seen failure except for the last few years with the launch of ROCKET and InceptionTime. Inspired by their contributions, this project endeavors to test various model architectures famously used in deep learning to this task after feature extraction using a relatively novel PPV feature vector first introduced in 2019, and thus frame a framework to develop future time series models on, agnostic to their primary task and data domain.

1 Introduction

Time series data is one of the most ubiquitously available data that can be potentially used for many tasks and insights that are otherwise unavailable to businesses and research communities. Being able to classify time series data is an important task in this research direction. Unfortunately, due to the high disparity between the train and testing split of many of the datasets available, there is a significant limit to the effectiveness of traditional deep learning methods in the field. Only recently InceptionTime [1] and ROCKET [2] were breakthrough architectures for this task, however still have not managed to match the performance of the state of the art in many datasets. With the goal of developing a unified framework for time series classification, prediction and other tasks, the project is shaped to showcase the feasibility of using a unique yet common feature vector instead of time series data for future model research in the time series domain.

Being able to develop a time series domain agnostic model is very similar to how Convolution Nets developed for images. This transition has always been a positive step for research and simultaneous operation that comes from this, is very valuable in many fields like stock market prediction, supply chain analytics, gesture recognition, and as the pandemic of COVID19 has proved, medical health service requirements as well. The ability to extract useful features and then feed the features to unique architectures for the two tasks of prediction and classification is a modular approach that many research groups endorse. For example, the ROCKET approach introduces the use of a novel feature called PPV or proportion of positive values, which is a unique new feature to build models around. The authors have used convolution operations with randomized kernels as a way to process the data before they feed it to a ridge classifier (for small datasets). This approach is unique and aids to the speed of the training using ROCKET. The feature PPV will play a major role in the models discussed in this project.

2 Prior Work

Traditionally, there are 3 ways to handle a given batch of time series data to be fed into various models. They can be listed as: 1) Naive Approach – Directly using the data as varying length, normalized

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

data 2) Dictionary Approach – Identifying repeating patterns in the data and using them as a feature. 3) Feature Generation – Processing the data using a method, followed by normalization and feeding to the network. The naive approach is useful when the data set is not large, is of a standard length, and the number of classes to classify into is low. This in essence raises the bar to beat for the more complex and sophisticated methods. The classification can be done using SVMs or ridge classifiers without too many parameters to tune.

The dictionary approach gained a lot of praise with the presentation of the BOSS algorithm [3]. The Bag of Words approach to time series yielded exceptional results but faced the challenge of memory and time constraints. The complexity and time complexity of the algorithm was very high and an issue to iron out. InceptionTime modified this approach. In essence, by using varying length kernels in 1D convolutions in an Inception Network style architecture, helped them extract short-term and long-term patterns within a sequence. This of course helped increase the performance of the dictionary approach and dramatically reduced the effective amount of time needed to train the classification models.

The feature generation approach involves feature engineering and processing the data with some distinct pre-processing function before the model. This includes Wavelet Transforms [4], Fourier Transforms, ROCKET feature transform, and more. All of these models were built using a data transform that extracted useful features from the time series data. Interestingly however, they also observed that it works better on some data sets than others. This was an interesting observation which led to the current state of the art, the HIVE-COTE algorithm [5]. In its essence, the HIVE COTE algorithm is an ensemble technique that uses different models for the same data set and using a voting scheme, chooses the best method for the given data set. This is a useful technique that is often employed in many deep learning tasks and has been the state of the art for the last 3 years. Interestingly, since ROCKET and InceptionTime were developed after the release of HIVE COTE, in some data sets, they have been able to overtake the algorithm in classification accuracy and completely overtake the HIVE COTE in terms of lower time complexity. These models showcased the latent ability and suitability of deep learning and neural networks in the field of time series classification and inspired this project.

3 Approach

3.1 ROCKET Kernels

The approach for this project was to iterate over different architectures using data generated after extracting useful features like the PPV from the time series data. The approach to calculate the PPV allows us to decide the number of kernels we want to use and thus gives us control over the effective size of the input to the models.

The main hyperparameters in the data preprocessing are the number of random kernels (k) and the number of random iterations (r). The effective size of the output is (2*k, r) for each example in the input time series, effectively making it a 2D matrix, irrespective of the length of the time series and the number of classes to classify into. This transform follows the procedure of random convolution kernels explained in ROCKET. In a nutshell, random non-trainable convolution kernels [6] are used with a normally distributed weight sequence, a uniformly distributed bias sequence, uniformly distributed (but controlled by number of kernels) dilation sequence and a chosen filter length from a fixed set of options. The output of the convolutions are computed as the max pooled value and the proportion of positive values. These two features are then aligned in a new feature vector of the size previously specified.

3.2 Models

In ROCKET this feature vector is directly fed to a Ridge Classifier as implemented in the popular python ML library, sci-kit-learn. This is where the similarity of ROCKET and the models presented in this project ends. The models instead use a more sophisticated network for classification. The architectures of the models were tested over multiple parameters to ensure the effect of each new property is understood carefully. Four architectures were designed, constructed and their performance were tested accordingly, and they are:

1) Basic Dense Network – A series of fully connected layers followed by a softmax classifier.

2) Residual Net Inspired Network – A series of ResNet blocks (projected blocks and 2 identity blocks) followed by global max pooling and fully connected layers.

3) Inception Net Inspired Network – A series of varying kernel convolution layers with skip connections for a deeper network followed by max pool and fully connected layers.

4) Auto Encoder Architecture – A dimension reduction network (encoder) followed by a split network trained on being able to re-generate the original time series.

The basic dense network was built across a varying number of dense layers. Each dense layer had a dropout layer, with a dropout strength of 0.3 to 0.5, associated to it for regularization and the units were only linearly activated. This decision was inspired by the fact that many major models, including ROCKET use linear activation to reduce the task into a regression problem. The final layer is activated by soft-max. No additional activity regularization or kernel regularization was applied here.

The Res Net inspired architecture has multiple blocks of 1 projected block and 2 identity blocks, with each sub-block have three convolution layers depth. The advantage of ResNet was always the additional depth that was accessible without the issue of vanishing gradients. This was a critical reason for implementing a version with the same fundamental units or blocks. The input feature vector of:

$$2 * numberofkernels$$

allowed the possibility of reshaping the input into a 2D image-like form that is convenient for 2D convolutions. The familiar structure and optimization methods were followed in the designing of the deep residual CNN, and number of filters varied from 200 to 1200 layer to layer. Before the fully connected layers, a global max pooling layer was used for dimensionality reduction and for making the model more robust to subtle changes in the weights before it. L2 regularization was applied for every convolution kernel and L1 regularization was applied for the bias term. The activation function this time throughout the model except the final layer was chosen to be the rectified linear unit or ReLU. This was done so because of the ability of the gradients to be easily scaled and carried throughout the network without an exploding gradient issue. The hyperbolic tangent activation function was also attempted.

The Inception Net inspired architecture tried to capture longer dependencies in the PPV by using varying length kernels. The model was not designed to be very deep, but broad instead. Convolution layers of kernel size 1, 3, 7, 11, 15, 19 were used in repeated fashion followed by a concatenate layer and max pooling. Some residual skip connections were also used to help with the vanishing gradient issue. Similar to the model previously described, L2 regularization was applied to the kernel and the bias for each convolution layer.

The auto encoder architecture was inspired by the goal of developing a two-loss model such that the auto-encoder makes class predictions after the encoder and tries to recreate the input feature vector after the decoder stage. This increases the training efficiency and the results advocate this as well for some of the datasets. The encoder stage was designed to be a sequence of convolution and max pooling layers, while the decoder was a sequence of transposed convolutions (often confused with de-convolution, which is incorrect). Categorical cross entropy was used for the output of the encoder and mean squared error was used for the decoder output. This meant there were two gradients being back-propagated which would have better optimized the weights for the classification task. This is a unique architecture for the time series classification task that was tried for the project.

Every other model was trained with a loss function of categorical cross-entropy. The optimizer chosen was Adam with a learning rate schedule of 0.001 for the first 20 epochs, followed by 0.0001 for the next 100 epochs and 0.00001 for the rest. The learning rate was also reduced further by a factor 0.5 every 10 epochs where the validation loss did not change or in other words, plateaued.To support the training, sufficient regularization through dropout, batch normalization and/or activity regularization have been included to a varying degree into all the architectures.

Table 1: Testing Data Sets from UCR Repository

Name	Train Set	Test Set	Length	Classes	Category
Abnormal Heartbeat	303	303	3053	5	Audio
Phoneme	214	1896	1024	39	Sound
Arrow Head	36	175	251	3	Image
Face All	560	1690	131	14	Image
Beef	30	30	470	5	Spectro
Ethanol Level	504	500	1751	4	Spectro
Strawberry	613	370	235	2	Spectro
Earthquake	322	139	512	2	Sensor
Inline Skate	100	550	1882	7	Motion

3.3 Data Set

For the datasets, the UCR database [7] was used. It contains all the recent state of the art benchmarking datasets. It is almost always referred to in time series classification, and the diversity of the datasets available there was certainly helpful in understanding the operation of the models. The unfortunate part of time series data, is the highly imbalanced train-test ratio that could potentially be an overfitting nightmare. The model architectures, learning rate, regularization parameters, and depth of the networks, were tuned on 15 datasets from the UCR dataset collection, and the following datasets were chosen as a diverse collection that will best represent the performance of the PPV parameter and the models. The datasets used for testing are described in table 1.

These datasets were chosen because of the varied categories they belong too, the varying train to test ratio for each dataset, allowed for a good demonstration of class weighting (to handle class imbalance), and common normalized features (max and PPV) after transformation using the same. The unfortunate problem as with any time series datasets, is that the distribution of classes may not be the same for the train, validation, and test sets. This is of course, one of the reasons of the fluctuations in the performances as the next few figures show. This could have been easily solved by pooling all the data together and splitting it again with a more standard distribution. Unfortunately, none of the publications have mentioned doing this, instead using this as a benchmarking dataset library. So, this step was omitted.

3.4 Metrics

Since this is a classification task, multiple metrics were used. True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN), Precision (P), Accuracy (A), Recall (R), and Area of Curve (AOC) were all calculated for the models and datasets. This was done to give a better understanding of the performance of the classifier as accuracy alone cannot be used as a metric to judge a classifier on. Interestingly however, most datasets only present the peak accuracy achieved by a model without much giving much attention to other performance and confidence parameters. This should evolve into a combination of Accuracy, F1-Score (a composite of the precision and recall) and Area of Curve Scores. This is because while the accuracy of a model can be reported as high, if the area of curve is closer to 0.5 or 50 percent, then the effective confidence in that accuracy is considerably lower. Similarly, the F-1 score by its own is considered to be futile as it heavily depends on the distribution of data, but by coupling it with the Accuracy and AoC of the model, we can have a more through understanding of the performance of the model.

4 Experiments

All models were trained using the same features extracted to compare the performance on relatively standardized parameters. The pipeline used was pretty straightforward as figure 1 shows.

Due to time and hardware constraints the number of epochs was fixed at 200, and the PPV transform parameters were fixed to number of random kernels as 10000. The option list for the length of the kernel length was limited to a list of 7, 9, and 11. Since the datasets vary across multiple domains, time lengths, and train to test ratios, the results are diverse which help us to understand the efficacy of

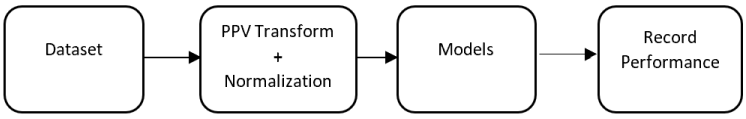

Figure 1: The Data Pipeline

the PPV parameter and design of the models better. All the training and testing was conducted locally using an Nvidia RTX 2070 SUPER GPU with 8 GB memory. The limited memory also enforced a limit on the batch size to 32.

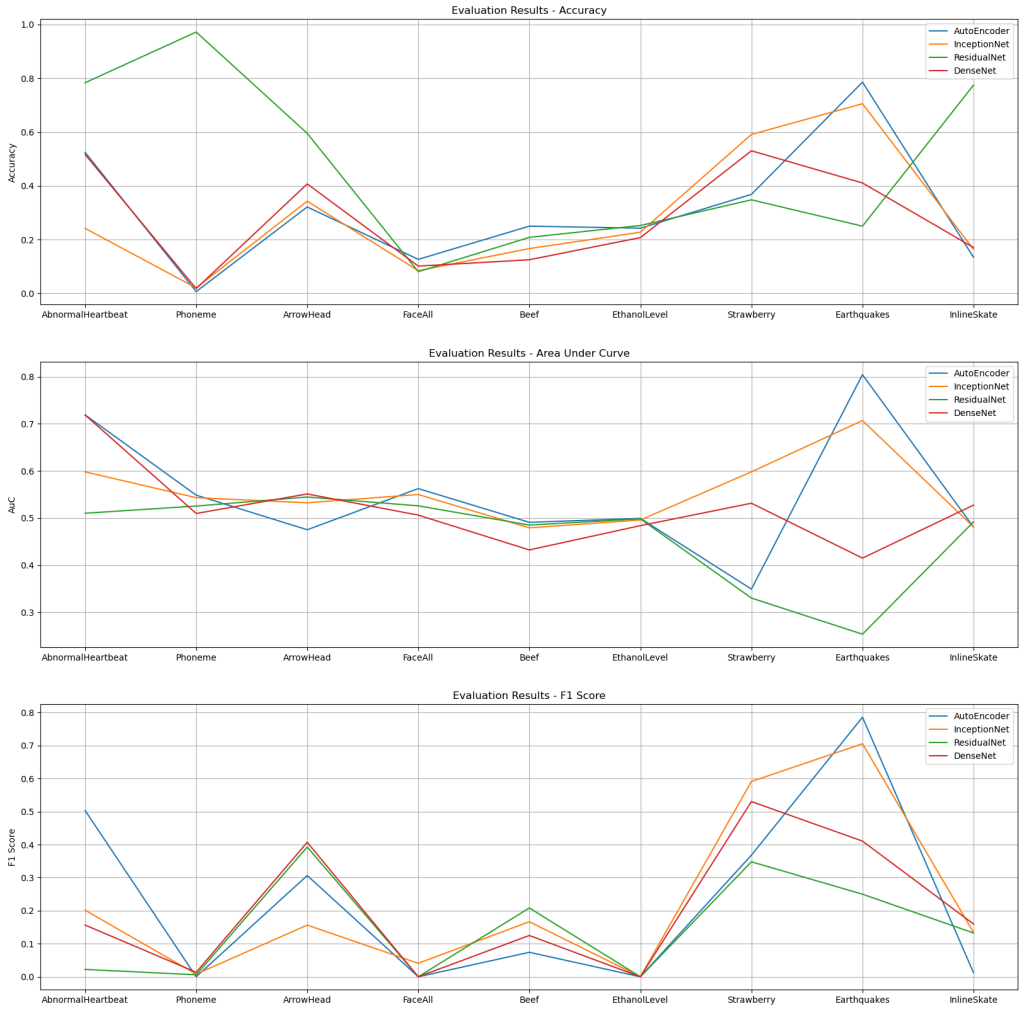

Figure 2: Evaluation Results

Figure 2 represents the evaluation results of the four models in the 9 data sets used for testing. The plotted metrics include Accuracy, Area Under the Curve (AUC) and F-1 Score to better understand the effect of the multiple models, and the various data sets. As the figure shows, in most scenarios the AutoEncoder architecture had performed the best with highest confidence levels. The interesting outlier would be the residual net performance in the phoneme data set. This would not be easily understood just by observing the accuracy maps. Coupling it with the AUC value clearly indicates low confidence the predictions. AUC can be interpreted as a value that helps us understand the confidence in predictions. A value closer to 0.5 indicates low confidences while a value closer to 1 shows the classifier is working as expected. Alternatively a value closer to 0 indicates the the classifier is working opposite to the way it was designed.

Figure 3 (top right) shows this with more clarity. There is strong evidence to suggest the Res Net model complexity is far higher than needed and the regularization is not sufficient. The high fluctuation in values also presents the a peek into the data. In the case of phoneme there is an highly imbalanced data set with poor train to test ratio and a large number of classes to predict from.

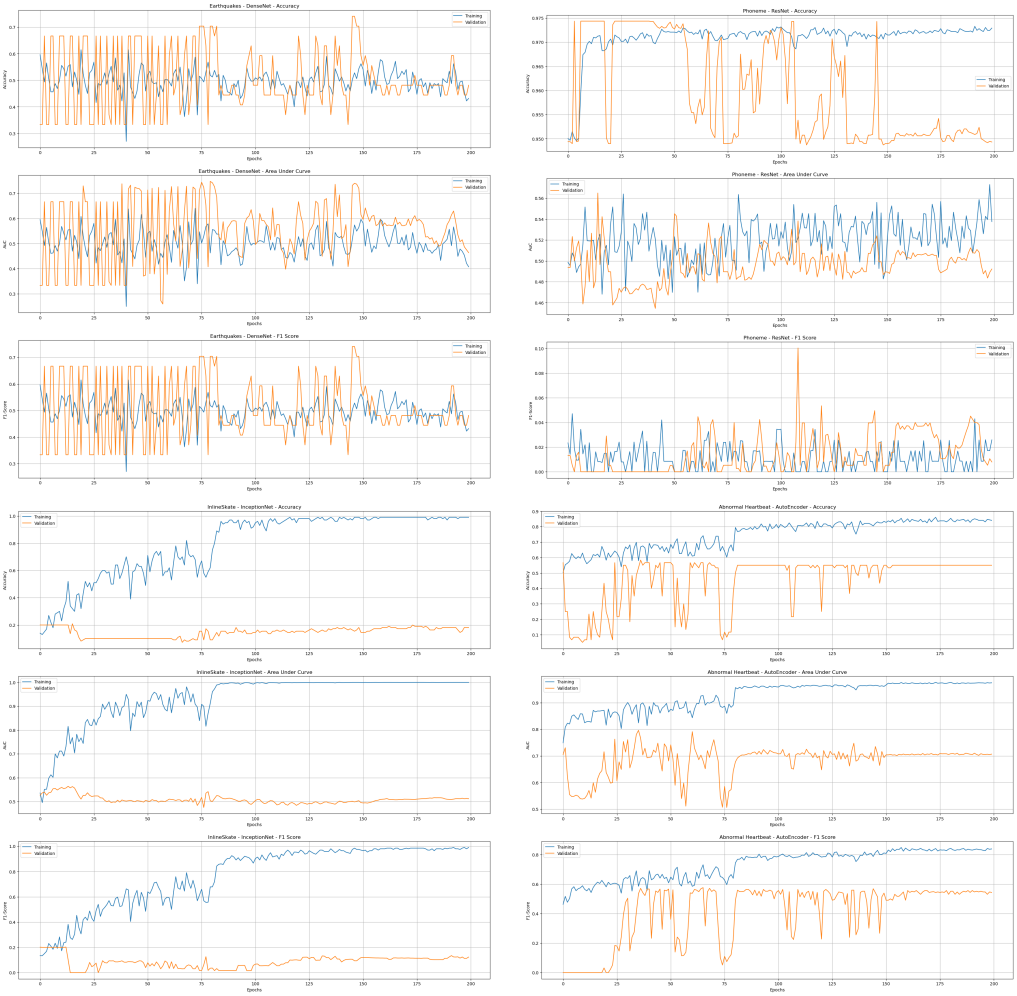

Figure 3: Model Performance with Individual Datasets - Dense Net vs Earthquake [Top Left], ResNet vs Phoneme [Top Right], Inception Net vs InlineSkate [Bottom Left], AutoEncoder vs Abnormal Heartbeat [Bottom Right]

The dense net performance showcased in figure 3 (top left plot) is highly volatile as well. Here the we can observe the dense net's performance on the earthquake data set; a data set with the best performance achieved by the Auto Encoder architecture. The highly fluctuating nature of the classifier here in the initial epochs is very likely due to a large learning rate, and limited number of tune-able parameters in the dense net. The F-1 score is fairly respectable at 0.5, but the model can be tuned/improved much further. It is very important to note that all these models have been trained on the feature vector generated from the randomized kernels. In that regard, this is very respectable performance.

Figure 3 (bottom left) shows the Inception Net performance and the need for regularization becomes very apparent. Inline Skate data set is another imbalanced data set but with fairly low number of parameters, extremely quick training (200 epochs in roughly 100 seconds). Here, while the training AUC and accuracy were closer to one, indicating very acceptable results. The validation accuracy and AUC are less than the training metrics, indicating overfitting. Since the model was suing less parameters, I suspect the model design was not an issue, but the regularization needed to be increased.

The autoencoder represented the best balance between training time, model size, highest average AUCs, and effective accuracy. Figure 3 (bottom right) shows the autoencoder performance with respect to the Abnormal Heartbeat data set. This result is very promising that gives further promise to the method of designing custom models beyond the feature extraction using the random kernels. More experimentation seems warranted after these results, especially its performance in the balanced, Earthquake data set.

5 Conclusions and Future Work

There were many interesting takeaways from this project and the resources and references referred to during the course of the project. Firstly, the ability to extract useful information from time series data without committing to one classification model helps dramatically in the process of developing models for the specific use case in mind. Being able to derive the PPV feature vector seems to be a major step forward in the development of time series classification models. The advantages to consider include fixed series models (and maybe one step closer to the true AlexNet of any time series data), shorter training time due to the significantly more flexible features and lastly, the size conversion of time series vectors to the PPV and max values. With more experimentation and research, it is considerably more open to progress and something that could be worked on in the future.

This unified framework for time series classification can also be stretched to the time series prediction domain, though due to time constraints, the feasibility of that was not completely verified. It could be yet another topic of further research. But in conclusion, it is clear that using PPV and randomized kernels is a step we can take for any time series data without major concerns of performance, as long as we design the rest of the classifier correctly. This project also demonstrates the effectiveness of the of using the AUC and F1 score metric in conjunction with the accuracy metric, which continues to be the de-facto standard models are compared on.

References

[1] Inception Time: Finding AlexNet for Time Seties Classification - Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Chalotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, Francois Petitjean - Data Mining and Knowledge Discovery (2020)

[2] ROCKET: Exceptionally Fast and Accurate Time Series Classification Using Random Convolution Kernels - Angus Dempster, Francois Petitjean, Geoffrey I. Webb - Data Mining and Knowledge Discovery (2020)

[3] The BOSS is Concerned with Time Series Classification in the Presence of Noise - Schafer P - Data Mining and Knowledge Discovery (2015)

[4] Binary Shapelet Transform for Multiclass Time Series Classification - Bostrom A, Bagnall A - Big Data Analytics and Knowledge Discovery (2015)

[5] Time series classification with HIVE-COTE: The Hierarchical Vote Collective of Transformation Based Ensembeles - Lines J, Taylor S, Bagnall A - ACM Transactions Knowledge Discovery and Data (2018)

[6] Random Projection Filter Bank for Time Series Data - Farahmand A, Pourazarm S, Nikovski D - Advances in Neural Information Processing Systems (2017)

[7] The UEA and UCR Time Series Classification Repository - Bagnall A, Lines J., Vickers W., Keogh E. - http://www.timeseriesclassification.com - 2019