Recurrence and Self-Attention vs the Transformer for Time-Series Classification: A Comparative Study

Alexander Katrompas[1], Theodoros Ntakouris[2], and Vangelis Metsis[1]

[1] Texas State University, San Marcos TX 78666, USA
{amk181,vmetsis}@txstate.edu
[2] University of Patras, Patras, Greece
ntakouris@ceid.upatras.gr

Abstract. Recently the transformer has established itself as the state-of-the-art in text processing and has demonstrated impressive results in image processing, leading to the decline in the use of recurrence in neural network models. As established in the seminal paper, Attention Is All You Need, recurrence can be removed in favor of a simpler model using only self-attention. While transformers have shown themselves to be robust in a variety of text and image processing tasks, these tasks all have one thing in common; they are inherently non-temporal. Although transformers are also finding success in modeling time-series data, they also have their limitations as compared to recurrent models. We explore a class of problems involving classification and prediction from time-series data and show that recurrence combined with self-attention can meet or exceed the transformer architecture performance. This particular class of problem, temporal classification, and prediction of labels through time from time-series data is of particular importance to medical data sets which are often time-series based. [3].

1 Introduction

The transformer architecture has shown superior performance to recurrent networks (RNN) and convolutional (CNN) networks, particularly in the areas of text translation and processing [13], as well as recently in image classification [17]. Self-attention only models, based on the transformer, are also showing promise in time-series classification [16]. While the majority of these non-temporal data (i.e. image and text) have some form of order, they are not inherently temporal nor continuous in nature. Text data and images are processed by transformers as discrete data tokens or patches where the value of one token does not necessarily affect the value of a neighboring token. While order and position matter in text prediction, the strength of the relationships between tokens is at least semi-independent of proximity and requires positional encoding for modeling. [13]. This applies to images as well [15].

Conversely, time-series data are typically continuous, and proximity is important to the current time-step and the current classification/prediction [9].

[3] Source code: https://github.com/imics-lab/recurrence-with-self-attention

This study will show it is this difference in time-series data characteristics which make recurrence combined with self-attention important to temporal classification/prediction.

Previous work has shown that time-series classification can benefit from the addition of self-attention to recurrent neural networks [6]. The main contribution of this paper is to demonstrate that in the case of temporal classification/prediction, neither self-attention nor recurrence is all you need, but rather it is recurrence combined with self-attention which provides the most robust modeling for this class of problems. The goal of this work is to compare and contrast self-attention alone, i.e. the transformer, against combined recurrence and attention to achieve the optimal time-series modeling. This is verified through a series of experiments on nine different publicly available data sets, empirical observations, and theoretical evidence.

2 Background

2.1 Recurrent Neural Networks

Recurrent networks are particularly adept at maintaining temporal information through the recurrence mechanism, which feeds the current recurrent layer's output back to input layer, thereby including each current output to the subsequent input and forming a temporal chain of causality by maintaining an internal state (i.e. "memory"). This architecture allows the RNN to more effectively model time-series data than most other networks [4]. The term "recurrent neural network" is used broadly to refer to a collection of specific network architectures. Of interest in this study is the LSTM which has proven itself as one of the most robust of the RNN architectures [10,4].

While RNNs have proven themselves in a variety of tasks, especially temporal modeling, they have some drawbacks including computational complexity and the heavier weighting of nearer time-steps [13,7,5]. The latter characteristic is of interest to this study. A RNN's "memory" has a time-dependent feature, which is both a shortcoming and a benefit. RNNs tend to weigh the most recent information more heavily than long past information, analogous to a weighted moving average. This is a benefit in modeling time-series data where more recent information should be weighed heavier [10]. However, it is also a shortcoming in two regards: 1) long past information is "forgotten" even when it is useful, and 2) within any particular sequence we may not always wish a weighting strictly based on proximity in time.

2.2 Self-Attention

Attention mechanisms, originally created for text prediction, allow the network to "attend" to portions of a sequence out of order, stressing the importance of one token or another within a sequence, thereby creating a better representation of the sequence [13,4]. There are several types of attention mechanisms, all of

which work similarly within a neural network to create a "relationship within a relationship"; the latter relationship being the neural network input/output and the former being some relation between the sequence tokens which are more (or less) important to the neural network prediction. Attention can be generally described as a weight or "context vector" of importance within a sequence [2,8,7].

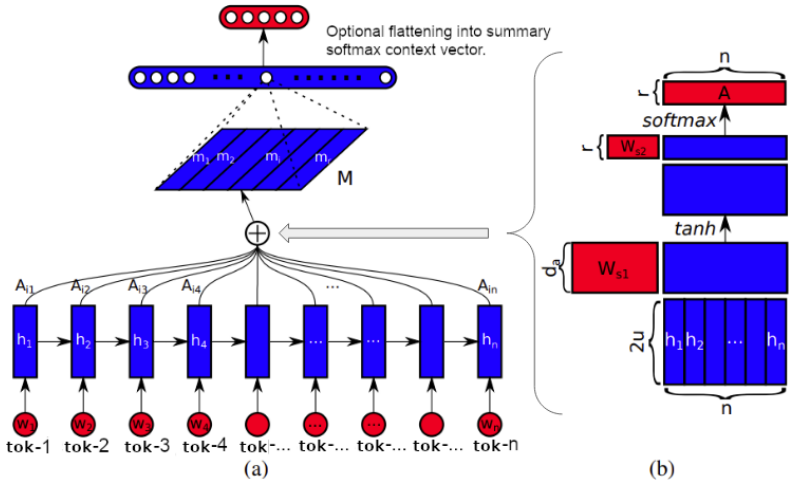


Fig. 1: Attention-based LSTM model (a) [7] with a self-attention layer (b).

Self-attention (see Figure 1) is an attention mechanism directly relating different positions of a sequence in order to compute a better representation of that sequence. Self-attention differs from other attention mechanisms in that rather than calculating an entire summarized context vector based on input/output prediction, self-attention is directly calculating sequence-portion importance relative to other sequence-portions [13,4,17,7].

2.3 LSTM with Self-Attention

When combined with LSTM architectures, attention operates by capturing all LSTM output within a sequence and training a separate layer to "attend" to some parts of the LSTM output more than others [7]. For an input sequence $\boldsymbol{x} = (x_1, x_2, ..., x_T)$ the LSTM layer produces the hidden vector sequence $\boldsymbol{h} = (h_1, h_2, ..., h_T)$ and output $\boldsymbol{y} = (y_1, y_2, ..., y_T)$ of the same length, by iterating the following equations from $t = 1$ to T.

$$h_t = \mathcal{H}(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \tag{1}$$

$$y_t = W_{hy}h_t + b_y \tag{2}$$

where the W terms denote weight matrices, the b terms denote bias vectors, and $\mathcal{H}$ is the hidden layer function. In this fashion, self-attention learns to weigh portions of a sequence for relative feature importance [15].

The attention used in this study is multiplicative self-attention and uses the following attention mechanism:

$$h_t = tanh(W_x x_t + W_h h_{t-1} + b_h) \tag{3}$$

$$e_t = \sigma(h_t^T W_a h_{t-1} + b_t), a_t = softmax(e_t) \tag{4}$$

where h_t is the hidden node output from the LSTM layer in a two-dimensional matrix. e_t is the sigmoid activation output of the attention two-layer network, where W_a is the attention network weights, producing a corresponding matrix of the attention network activations. a_t is the softmax activation of e_t producing a context vector "alignment score" weighing the importance of the sequences [6].

2.4 Transformer

The transformer is a deep learning model which primarily uses the self-attention mechanism for modeling and eliminates the RNN component. Like RNNs, transformers are designed to handle ordered input data (e.g text, images) for tasks such as translation, summarization, prediction, and classification. Unlike RNNs, transformers do not process data sequentially at the encoder (input) stage. Instead, the transformer processes the tokens in parallel and identifies the context of each token relative to other tokens, which then confers meaning to each word in the sentence [13].

Transformers typically adopt an encoder-decoder architecture. The function of the encoder layers is to generate encodings that contain information about which parts of the inputs are relevant to each other. Each decoder layer does the opposite, taking all the encodings and using their incorporated contextual information to generate an ordered output sequence. To achieve this, each encoder and decoder layer makes use of a self-attention mechanism [13]. One or more fully connected layers can be attached at the end of the encoder part of the transformer to create a sequence classification architecture.

Because transformers do not process data in order at the encoder (input) layer, they do not understand order on their own (which RNNs understand by design). To solve this, transformers use positional encoding to maintain order in output (discussed further in sections 3.1) [13].

2.5 Temporal Classification and Prediction

Text and image processing require ordered information and relative position matters, however, strict sequential ordering and proximity do not matter (or at least matter less) [2,13]. Conversely, time-series data are highly dependent on absolute order, proximity, and absolute position [9]. Further, while text and image data are typically processed in discrete "chunks" such as words or patches, time-series data are typically processed as a series of continuous signal measurements in which the strongest relationships to any given time-step are with the immediately preceding time-steps.

Given the characteristics of text/image data versus time-series it can immediately be seen why the transformer has enjoyed such success with text and image processing. Not only is recurrence not needed, but it can also be argued it is counter productive to use an architecture based on continuous temporal flow for such tasks. By that same logic, it is also fair to question the transformer's suitability for true time-series data and temporal classification, even considering recent promising advances [16]. To do so we first identify an important and common class of temporal classification and pattern matching problem which is highly time-dependent.

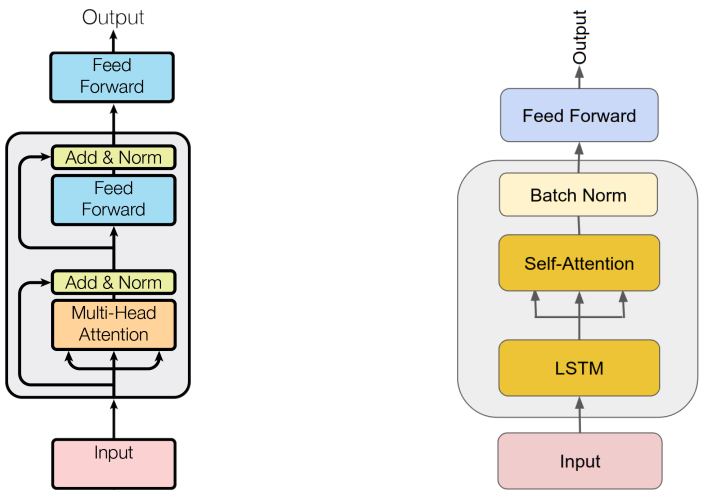
The time-series problem in question is the classification or prediction of a temporal "event" (i.e. label) through time. For example, given a continuous ECG signal can we identify heart abnormalities, or given accelerometer readings from a smart device can we detect activities of daily life (ADLs), and can we do so with superior results to both RNNs alone and the transformer. Such tasks require pattern matching and modeling data which have three very specific characteristics. 1) *Sequential nature*: The data to be modeled is true time-series data, continuous and in-order, sampled at reasonably regular rates. 2) *Natural order*: The data to be modeled are natural and not artificially staged into fixed, discrete, disparate labels. 3) *Temporal label classification from time-series data*: The classification labels are occurrences through time from time-series data and not single-point, discrete classifications.

3 Models

3.1 Time-Series Transformer

Architecture: The time-series transformer used in this study is built directly from the transformer described in [13] with a modified output for time-series classification, and eliminating positional encoding as it is not needed (see section 3.1). The self-attention mechanism is identical to [13], as is the encoder architecture, including layer normalization and feed-forward components. Since decoding is not needed, the decoder is replaced with dense layers, leading to the final dense layer for output. Figure 2a illustrates this architecture.

Positional Encoding: In the traditional encoder-decoder transformer relative and/or absolute positional encoding is needed to preserve order within sequences and in output [13,12], and to signify the absolute position of each token in the sequence, as the same token can appear in different positions of a sequence for different samples. However, in time series where each token is a single time-step, i.e a signal measurement appearing as a scalar real number or a vector (in the case of multiple channels), positional encoding does not add any information. The same real number may never appear again in the dataset or it may appear multiple times within the same sequence. Thus adding a positional encoding to time-series data only adds another feature to be learned from the training data, which does not add any benefit to the prediction performance. This is verified in practice, as in all our experiments attempting positional encoding, accuracy declined between 1% and 4% regardless of hyper-parameter tuning.



(a) Time-series transformer architecture. (b) LSTM with self-attention architecture.

Fig. 2: Overview of neural network architectures evaluated in this study.

3.2 LSTM with Self-Attention

Architecture: The basis for an LSTM plus self-attention model is that we wish to model sequences in order through time (the LSTM component) while also attending to portions of a sequence which may be more or less relevant (self-attention), thereby making a better representation of each sequence. In this way, we are modeling ordered temporal sequences which are better representations of themselves through the attention mechanism, while at the same time allowing the LSTM memory to remember and model time-series dependencies. Figure 2b illustrates the overall LSTM/self-attention architecture.

Figure 3 illustrates the conceptual flow of data "weight" through both a LSTM model (top) and a LSTM plus attention model (bottom). Darker shading indicates a stronger relationship to LSTM memory and/or attention. As data flows through an LSTM more recent data is "weighted" heavier than past data. The top flow (LSTM alone) illustrates that data is weighted smoothly both intra-sequence and inter-sequence.

The lower flow of figure 3 shows LSTM plus self-attention with each sequence modeled according to self-attention. Within a sequence, we can see some time-steps as more or less important to the sequence overall. As stated in [13], through self-attention we can "*form a better overall representation of the sequence.*"

Sequence Length: Deep learning architectures require the continuous time series to be broken down into fixed-length sequences for training. Sequence length is a hyper-parameter which can be tuned on a per-model basis. Model accuracy is sensitive to sequence length selection which makes both intuitive and theoretical sense. When modeling time-series data with a self-attention component we are attending more or less to portions of a sequence. Therefore if we make the

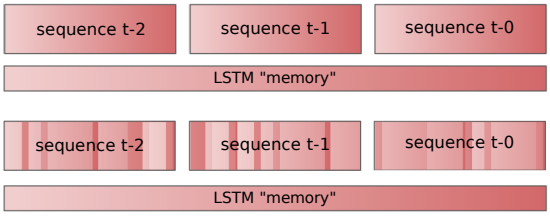


Fig. 3: Conceptual data flow through LSTM (top) versus LSTM plus self-attention (bottom). Darker shading indicates higher importance.

sequence very long then the model essentially degrades to attention only due to the LSTM's decreased ability to model time-steps too far in the past. In the reverse case, imagine the sequence length being simply 1 or 2 time-steps. In that case, there is not enough information for self-attention to intelligently attend to portions of sequence, and the model degrades to LSTM-only. Therefore, as sequence length is lengthened or shortened, more or less emphasis is placed on the self-attention mechanism versus the LSTM. For this study sequence length was selected through grid search.

4 Experiments

Set-up: In each experiment, hyper-parameters were selected through grid-search and are noted in the referenced code. Sequence lengths, being of particular note, were selected as follows: SmartFall: 50, MobiAct: 200, Australian National Weather Observations: 5, Air Quality: 4, mHealth: 200, ECG: 100.

Data Sets: Data sets were chosen with the following rationale: (i) Satisfying the criteria of section 2.5. (ii) Verification and comparison to similar LSTM-only studies with ADLs and time-series data [6,14]. (iii) ADL data sets, and ECG sets, representing medical data. (iv) Sufficiently different classification tasks which bear little to no resemblance to each other, other than the fact they satisfy the criteria of section 2.5. (v) A mix of temporal classification (SmartFall, MobiAct, mHealth, EEG) and temporal prediction (Australian BOM, CO prediction). (vi) A mix of data sizes from 100s to 100,000s of instances. (vii) Both binary classification/predication and multi-label classification.
The data sets used are the following:

- *mHealth* (multi-label classification): Human behavior recording, analysis, and classification based on multi-modal body sensing [3].
- *ECG Heartbeat Categorization* (multi-label classification): Two collections of heartbeat signals derived from two datasets in heartbeat classification, the MIT-BIH Arrhythmia Dataset[4] and The PTB Diagnostic ECG Database[5].

[4] https://www.physionet.org/content/mitdb/1.0.0/
[5] https://www.deepdyve.com/lp/de-gruyter/nutzung-der-ekg-signaldatenbank-cardiodat-der-ptb-ber-das-internet-uemKpjIFzM

- *SmartFall* (binary classification): The data set consists of raw (x, y, z) accelerometer readings representing activities of daily living (ADLs), such as walking and running with falls interspersed [6]. The task is to label a fall as compared to other ADLs.
- *MobiAct* (binary classification): The data set consists of raw (x, y, z) accelerometer readings with various ADLs recorded and labeled [14,6]. The task is to label a fall, jogging, walking up stairs, walking down stairs, as compared to other ADLs.
- *Australian BOM National Weather Observations* (binary prediction): Observations of a number of weather elements each day in various Australian cities for years 2008 to 2017. The task is to predict rain/no-rain tomorrow [1].
- *Air Quality Time-Series data UCI* (binary prediction): Hourly averaged responses from an array of 5 metal oxide chemical sensors embedded in an air quality device [11]. The task is to predict if CO levels will rise/fall tomorrow.

The larger data sets were split into train, validation, and test 60/20/20. The smaller sets are split into train and test 80/20. For the data sets with validation sets, each model was run ten times and test set averages were calculated and recorded. In the case of smaller sets without validation sets, each model was run with 5-fold cross-validation and test set averages were calculated and recorded.

4.1 Results

All binary classification/prediction task results are presented in Table 1. The macro-averaged results for all classes of mHealth and EEG data set are shown in Tables 2 and 3. The detailed per-class results can be found in Appendix A. LSTM only results have been conducted on the same and similar data sets in other work [6,14]. In all cases and without exception, LSTM plus self-attention outperformed stand-alone LSTM models. Therefore, results for stand-alone LSTM are omitted for brevity since they do not alter the results of this study. Further, as shown in detail in this study, LSTM plus self-attention was generally superior to self-attention alone (i.e. the transformer).

Accuracy of the LSTM plus self-attention model showed an increase in validation accuracy over the transformer between 1.2% (MobiAct jog detection) and 14.4% (CO prediction). Precision increased from 1.4% (MobiAct jog detection) to 324.5% (SmartFall fall detection). Recall ranged from a decrease of 3.7% (MobiAct stairs up detection) to an increase of 8.7% (SmartFall fall detection). F1 increased from 0.7% (MobiAct jog detection) to 53.7% (SmartFall fall detection). Further, in all cases except one (Mobi-Act Jog), the LSTM plus self-attention architecture ROC-AUC and PR-AUC showed significant improvement over the transformer architecture.

For the mHealth data set, the LSTM/Attn model proved superior for all but one category, with overall accuracy being .85 versus .82 and a weighted average .87 versus .82. In the case of EEG classification, both models achieve extremely high accuracy, 0.99, however, the detailed per category results still show a very slight advantage to the LSTM/Attn model.

Table 1: Results comparing performance metrics of LSTM+self-attention against the Transformer for each *binary classification* dataset. Acc = Accuracy, Prec = Precision, Rec = Recall, F1 = F1-score, ROC=Area under ROC curve, PR = Area under Precision-Recall curve. SF = SmartFall, MA-[F/J/Up/Dn] = MobiAct [Fall/Jog/StairsUp/StairsDown], Rain = Australian BOM, CO = Carbon Monoxide.

	SF		MA-F		MA-J		MA-Up		MA-Dn		Rain		CO	
	tran	lstm	tran	lstm	tran	lstm	tran	lstm	tran	lstm	tran	lstm	tran	lstm
Acc	.886	**.961**	.900	**.937**	.957	**.968**	.894	**.936**	.900	**.916**	.848	**.923**	.659	**.754**
Prec	.196	**.832**	.622	**.767**	.976	**.990**	.868	**.972**	.891	**.956**	.486	**.827**	.422	**.698**
Rec	.782	**.869**	.913	**.953**	.977	**.975**	.967	**.932**	**.964**	.790	**.847**	.927	.651	**.715**
F1	.313	**.850**	.740	**.850**	.976	**.983**	.914	**.952**	.926	**.941**	.601	**.837**	.512	**.706**
ROC	.593	**.906**	.802	**.878**	**.872**	.868	.906	**.919**	**.906**	.888	.722	**.890**	.628	**.747**
PR	.542	**.862**	.811	**.887**	.987	.987	.960	**.961**	**.966**	.957	.699	**.858**	.659	**.771**

Table 2: Averaged results of mHealth data set (13 classes). Acc: Accuracy, Prec: Precision, Rec: Recall, F1: F1-score, MacrAvg: Macro Average, WAvg: Weighted Avg.

	LSTM w/Attn			Transformer			
	Prec	Rec	F1	Prec	Rec	F1	Support
MacrAvg	**0.76**	**0.83**	**0.78**	0.71	0.68	0.67	202k
WAvg	**0.87**	**0.85**	**0.85**	0.82	0.82	0.82	
Acc	**0.85**			0.82			

Table 3: Averaged results of ECG Classification data set (5 classes).

	LSTM w/Attn			Transformer			
	Prec	Rec	F1	Prec	Rec	F1	Support
MacrAvg	0.93	**0.99**	0.95	0.93	0.97	0.95	21.9k
WAvg	0.99	0.99	0.99	0.99	0.99	0.99	
Acc	0.99				0.99		

5 Conclusion

The LSTM plus self-attention model outperformed and proved superior to the time-series transformer in temporal classification and prediction. In particular, we believe this work shows that a combination of overall temporal modeling (the RNN component) along with fine-grained sequence modeling (self-attention) addresses this specific class of problem with state-of-the-art results. We believe this is notable since this particular class of problem, natural time-series data and temporal classification/prediction, is common and of particular importance to tasks such as medical data analysis and prediction, ADL classification, process control, natural event prediction, financial and economic modeling, and many other similar high-value tasks. This work shows recurrence is not only valuable but that it is a particularly and specifically valuable partner to self-attention, demonstrating that for this particular class of problem, attention may not be all you need.

References

1. Australian Bureau of Meteorology (BOM): Australia, Rain Tomorrow. Australian BOM National Weather Observations 8
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. ArXiv **1409** (09 2014) 3, 4
3. Banos, O., Villalonga, C., García, R., Saez, A., Damas, M., Holgado-Terriza, J., Lee, S., Pomares, H., Rojas, I.: Design, implementation and validation of a novel open framework for agile development of mobile health applications. BioMedical Engineering OnLine **14**, S6 (08 2015) 7
4. Cheng, J., Dong, L., Lapata, M.: Long short-term memory-networks for machine reading. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 551–561 (01 2016) 2, 3
5. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**, 1735–80 (12 1997). https://doi.org/10.1162/neco.1997.9.8.1735 2
6. Katrompas, A., Metsis, V.: Enhancing lstm models with self-attention and stateful training. In: Intelligent Systems and Applications. IntelliSys 2021. vol. 294. Springer, Cham. (2021) 2, 4, 7, 8
7. Lin, Z., Feng, M., Dos Santos, C., Yu, M., Xiang, B., Zhou, B., Bengio, Y.: A structured self-attentive sentence embedding (03 2017) 2, 3
8. Luong, M.T., Pham, H., Manning, C.: Effective approaches to attention-based neural machine translation (08 2015) 3
9. Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., Cottrell, G.: A dual-stage attention-based recurrent neural network for time series prediction (04 2017) 1, 4
10. Rahman, L., Mohammed, N., Al Azad, A.K.: A new lstm model by introducing biological cell state. In: 2016 3rd International Conference on Electrical Engineering and Information Communication Technology (ICEEICT). pp. 1–6 (2016) 2
11. Saverio De Vito: Air quality data set, https://archive.ics.uci.edu/ml/datasets/Air+quality 8
12. Shaw, P., Uszkoreit, J., Vaswani, A.: Self-attention with relative position representations. pp. 464–468 (01 2018) 5
13. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, L., Polosukhin, I.: Attention is all you need. 31st Conference on Neural Information Processing Systems (NIPS 2017) (06 2017) 1, 2, 3, 4, 5, 6
14. Vavoulas., G., Chatzaki., C., Malliotakis., T., Pediaditis., M., Tsiknakis., M.: The mobiact dataset: Recognition of activities of daily living using smartphones. In: Proceedings of the International Conference on Information and Communication Technologies for Ageing Well and e-Health. pp. 143–151. SciTePress (2016) 7, 8
15. Wang, J., Yang, Y., Mao, J., Huang, Z., Huang, C., Xu, W.: Cnn-rnn: A unified framework for multi-label image classification. 2016 IEEE Conference on Computer Vision and Pattern Recognition (04 2016) 1, 3
16. Wu, N., Green, B., Ben, X., O'Banion, S.: Deep transformer models for time series forecasting: The influenza prevalence case (2020) 1, 5
17. Zhao, H., Jia, J., Koltun, V.: Exploring self-attention for image recognition. pp. 10073–10082 (06 2020) 1, 3

A Appendix: Detailed Classification Report Results

Table 4: Experimental results mHealth data set. Cat: Category, Acc: Accuracy, Prec: Precision, Rec: Recall, F1: F1-score, MacAvg: Macro Average, WAvg: Weighted Avg

LSTM w/Attn					Transformer				
Cat	Prec	Rec	F1	Sup	Cat	Prec	Rec	F1	Support
Null	**0.94**	0.86	**0.89**	135,542	Null	0.88	**0.88**	0.88	135,542
Standing	0.60	0.98	0.74	6,144	Standing	**0.71**	0.98	**0.82**	6,144
Sitting	**0.72**	**1.00**	**0.84**	6,144	Sitting	0.71	0.89	0.79	6,144
Laying	0.92	0.98	0.95	6,144	Laying	0.92	0.98	0.95	6,144
Walking	**0.50**	**0.91**	**0.65**	6,144	Walking	0.47	0.44	0.45	6,144
Stairs-Up	**0.71**	0.48	0.57	6,144	Stairs-Up	0.64	**0.76**	**0.69**	6,144
Bends	0.89	**0.90**	**0.90**	5,274	Bends	0.89	0.65	0.75	5,274
Arm elev	**0.79**	0.57	**0.66**	4,864	Arm elev	0.70	0.57	0.63	4,864
Knee bend	0.66	0.49	0.56	5120	Knee bend	0.83	0.54	0.66	5,120
Cycling	0.51	0.79	0.62	6144	Cycling	**0.75**	**0.81**	**0.78**	6,144
Jogging	**0.88**	**0.89**	**0.89**	6144	Jogging	0.71	0.39	0.51	6,144
Running	**0.83**	**1.00**	**0.91**	6144	Running	0.55	0.9	0.68	6,144
Jumping	**1.00**	**0.95**	**0.97**	2,048	Jumping	0.46	0.05	0.09	2,048
MacAvg	**0.76**	**0.83**	**0.78**	202,000	MacAvg	0.71	0.68	0.67	202,000
WAvg	**0.87**	**0.85**	**0.85**	202k	WAvg	0.82	0.82	0.82	202k
Acc	**0.85**	202,000			Acc	0.82	202,000		

Table 5: Experimental results ECG Classification data set. Cat: Category, Acc: Accuracy, Prec: Precision, Rec: Recall, F1: F1-score, MacAvg: Macro Average, W.Avg: Weighted Average, 0: Non-Ectopic, 1: Superventrical Ectopic, 2: Ventricular Ectopic, 3: Fusion, 4: Unknown

LSTM w/Attn					Transformer				
Cat	Prec	Rec	F1	Sup	Cat	Prec	Rec	F1-score	Support
Cat	Prec	Rec	F1	Support	Cat	Prec	Rec	F1	Support
0	1.00	0.99	1.00	18,118	0	1.00	0.99	1.00	18,118
1	0.82	**1.00**	**0.90**	556	1	0.82	0.87	0.84	556
2	1.00	0.97	0.98	1,448	2	0.95	0.99	0.97	1,448
3	0.81	**1.00**	0.90	162	3	**0.87**	0.99	**0.93**	162
4	1.00	0.99	1.00	1,616	4	1.00	0.99	1.00	1,616
MacAvg	0.93	**0.99**	0.95	21.9k	MacAvg	0.93	0.97	0.95	21900
WAvg	0.99	0.99	0.99	21.9k	WAvg	0.99	0.99	0.99	21.9k
Acc	0.99	21.9k			Acc	0.99	21.9k		