 doi:10.20944/preprints202210.0224.v1

*Article*

# Ensemble of Networks for Multilabel Classification

**Loris Nanni[1], Luca Trambaiollo[1], Sheryl Brahnam[2], Xiang Guo[2] and Chancellor Woolsey[2]**

[1] University of Padova, Via Gradenigo 6, Padova, Italy; loris.nanni@unipd.it; luca.trambaiollo@studenti.unibo.it
[2] Missouri State University, USA; SBrahnam@missouristate.edu; XiangGuo@missouristate.edu; Chancellor1224@live.missouristate.edu
* Correspondence: loris.nanni@unipd.it;

**Abstract:** Multilabel learning goes beyond standard supervised learning models by associating a sample with more than one class label. Among the many techniques developed in the last decade to handle multilabel learning best approaches are those harnessing the power of ensembles and deep learners. This work proposes merging both methods by combining a set of gated recurrent units, temporal convolutional neural networks, and long short-term memory networks trained with variants of the Adam optimization approach. We examine many Adam variants, each fundamentally based on the difference between present and past gradients, with step size adjusted for each parameter. We also combine Incorporating Multiple Clustering Centers and a bootstrap-aggregated decision trees ensemble, which is shown to further boost classification performance. In addition, we provide an ablation study for assessing the performance improvement that each module of our ensemble produces. Multiple experiments on a large set of datasets representing a wide variety of multilabel tasks demonstrate the robustness of our best ensemble, which is shown to outperform the state-of-the-art. The MATLAB code for generating the best ensembles in the experimental section will be made available at https://github.com/LorisNanni.

**Keywords:** multilabel; ensemble; incorporating multiple clustering centers; gated recurrent neural networks; temporal convolutional neural networks; long short-term memory

## 1. Introduction

Multilabel learning extends the standard supervised learning model that associates a sample with a single label by simultaneously categorizing samples with more than one class label. In the past, multilabel learning has been successfully implemented in many different domains [1] such as bioinformatics [2-5], information retrieval [6, 7], speech recognition [8, 9], and online user reviews with negative comment classification [10, 11].

As proposed by various researchers, one of the most intuitive handlings of multilabel classification is to treat it as a series of independent two-class binary classification problems, known as Binary Relevance (BR) [12, 13]. However, this approach has several limitations: the performance is relatively poor, it lacks scalability, and it cannot retain label correlations. Researchers have tried improving these issues using chains of binary classifiers [14], feature selections [15-17], class dependent and label specific features [18], and data augmentation [6]. Among these investigations, the augmentation method in [6] has demonstrated top performance on some multilabel datasets using Incorporating Multiple Cluster Centers (IMCC). In biomedical datasets, hML-KNN [19] has also proven to be one of the best multilabel classifiers, its success relying on feature-based and neighbor-based similarity scores.

Generating ensembles of classifiers is yet another robust and dependable method for enhancing performance on multilabel data [16, 20], with bagging shown to perform well in several multilabel classification benchmarks [21, 22]. However, a common problem in bagging is pairwise label correlation. To solve this problem, [23] invented a stacking technique where binary classifiers were trained on each label. The resulting predictions of the classifiers became inputs into stacked combinations combined with a meta-level classifier

 doi:10.20944/preprints202210.0224.v1

whose output produced a final decision. Another algorithm of note is the RAndom k-labELsets (RAkEL) [20], which constructed ensembles by training single-label learning models on random subsets of labels. The reader is referred to [24] and [25] for a discussion of some recent RAkEL variants.

Rather quickly, multilabel systems incorporating deep learners have risen to the top in classification performance. The ascendancy of deep learning in this field is evident in the large number of open source and commercial APIs currently providing deep learning solutions to multilabel classification problems. Some open source APIs are DeepDetect [26], VGG19 (VGG) [27], Inception v3 [28], InceptionResNet v2 [29], ResNet50 [30], MobileNet v2 [31], YOLO v3 [32], and fastai [33]. Some of the commercially available APIs include Imagga [34], Wolfram Alpha's Image Identification [35], Clarifai [36], Microsoft's Computer Vision [37], IBM Watson's Visual Recognition [38], and Google's Cloud Vision [39]. A comparison of performance across several multilabel benchmarks is reported in [40].

This paper proposes an ensemble for multilabel classification that combines Long Short-Term Memory networks (LSTM), Gated Recurrent Units (GRU) [41], and Temporal Convolutional Neural Networks (TCN) [42]. We posted some early preliminary results using this approach on ArXiv [43]. A GRU can be conceptualized as a simplified Bidirectional Long Short-Term Memory (LSTM) model. Both GRU and LSTM have hidden temporal states with some gating mechanisms. Both networks suffer from intermediate activations that are a function of low-level features. To offset this shortcoming, these models must be combined with classifiers that discern high-level relationships. In this work, we investigate TCN as a complement since it offers the advantage of hierarchically capturing relationships across high, intermediate, and low-level timescales.

Diversity in ensembles composed of LSTM, GRU, and TCN is assured in our approach by incorporating different Adam optimization [44] variants. Adam is well-known for finding low minima of the training loss, with many variants augmenting Adam's strengths and offsetting its weaknesses. As will be demonstrated, combining ensembles of LSTM, GRU, and TCN with IMCC [6] and a bootstrap-aggregated (bagged) decision trees ensemble (TB) [45] (carefully modified for managing multilabel data) further enhances performance.

Some of the contributions of this study are the following. To the best of our knowledge, we are the first to propose an ensemble method for managing multilabel classification based on combining sets of LSTM, GRU, and TCN, and we are the first to use TCN on this problem. Two new topologies of GRU and TCN are also proposed here, as well as a novel topology that combines the two. Another advance in multilabel classification is the application of variants of Adam optimization in our ensemble approach. Finally, for comparison with future works by other researchers in this area, the MATLAB source code for this study is available at https://github.com/LorisNanni.

The effectiveness and strength of our approach are demonstrated in the experimental section by comparing the performance of different ensembles with some baseline approaches across several multilabel benchmarks. Our best ensemble is compared with the best multilabel methods tested to date and shown to obtain state-of-the-art performance across many domains.

This paper is organized as follows. In section 2, we report on some recent work applying deep learning to multilabel classification and describe the benchmarks and performance indicators used in the experimental section. In section 3, each element and process in the proposed approach is detailed. This section covers a brief discussion of the preprocessing methods used and descriptions of GRU, TCN, IMCC, and LSTM networks. This section also describes pooling, training, and our method for generating the ensembles. In section 4, Adam optimization and all the Adam variants tested here are addressed. In section 5, experimental results are presented and discussed. Finally, in section 6, we summarize the results and outline some future directions of research.

 doi:10.20944/preprints202210.0224.v1

## 2. Related Work

One of the first works to apply deep learning to the problem of multilabel classification is [46]. The authors in that study proposed a simple feed-forward network to handle the functional genomics problem in computational biology. A growing body of research has since ensued that has advanced the field and application of deep learning to a wide range of multilabel problems. In [47], for example, a Convolutional Neural Network (CNN) combined with data augmentation was designed to tackle the problem of land cover scene categorization. Another CNN in [48] was developed to detect heart rhythm/conduction abnormalities. In that study, each element in the output vector corresponded to a rhythm class. CNN was combined with LSTM in [49] to handle the multilabel classification problem of protein-lncRNA interactions [49], and in [3] an ensemble of LSTM combined with a set of classifiers based on Multiple Linear Regression (MLR) was generated to predict a given compound's Anatomical Therapeutic Chemical (ATC) classifications. Recurrent CNNs (RCNNs) have recently been evaluated in many multilabel problems, including identifying surgical tools in laparoscopic videos [50] and in recommendation systems for prediagnosis support [51].

A growing number of researchers have also explored the benefits of adding GRUs to enhance the performance of multilabel systems. In [52], for example, sentiment in tweets were analyzed by extracting topics with a C-GRU (Context-aware GRU). In [53], a classifier system called NCBRPred was designed with bidirectional GRUs (BiGRUs) to predict nucleic acid binding residues based on the multilabel sequence labeling model. The BiGRUs were selected to capture the global interactions among the residues. In [54], an Inception model was combined with GRU network to identify nine classes of arrhythmias. See [55] for a recent survey for a deeper review of deep learning in multilabel classification.

### *2.1. Data Sets*

The following multilabel data sets were selected to evaluate our approach. These are standard benchmarks in multilabel research and run the gamut of typical multilabel problems (music, image, biomedical, and drug classifications). The names provided below are not necessarily those reported in the original papers but rather those commonly used in the literature.

1. Cal500 [56]: this dataset contains human-generated annotations, which label some popular Western music tracks. Tracks were composed by 500 artists. Cal500 has 502 instances, including 68 numeric features and 174 unique labels.
2. Scene [12]: this dataset contains 2407 color images. It includes a predefined training and testing set. The images are divided into six categories: beach (369), sunset (364), fall foliage (360), field (327), mountain (223), and urban (210). Sixty-three images have been assigned two category labels and one image three, making the total number of labels fifteen. The images all went through a preprocessing procedure. First, the images were converted to the CIE Luv space, which is perceptually uniform (close to Euclidean distances). Second, the images were divided into a 7×7 grid, which produced 49 blocks. Third, the mean and variance of each band were computed. The mean represents a low-resolution image, while the variance represents computationally inexpensive texture features. Finally, the images were transformed into a feature vector (49×3×2 = 294 dimensions).
3. Image [57]: this dataset contains 2,000 natural scene images. Images are divided into five base categories: desert (340 images), mountains (268 images), sea (341 images), sunset (216 images), and trees (378 images). Categorizing images into these five basic types produced a large set of images that belonged to two categories (442 images) and a smaller set that belonged to three categories (15 images). The total number of labels in this set, however, is 20 due to the joint categories. All images went through similar preprocessing methods as discussed in [12].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

4. Yeast [2]: this dataset contains biological data. In total there are 2417 micro-array expression data and phylogenetic profiles. They are represented by 103 features and are classified into 14 functional classes. A gene can be classified into more than one class.
5. Arts [6]: this dataset contains 5,000 art images, which are described by a total of 462 numeric features. Each image can be classified into 26 classes.
6. Liu [16]: this dataset contains drug data used to predict side effects. In total it has 832 compounds. They are represented by 2892 features and 1385 labels.
7. ATC [58]: this dataset contains 3883 ATC coded pharmaceuticals. Each sample is represented by 42 features and 14 classes.
8. ATC_f: this dataset is a variation of the ATC data set described above. In this dataset, however, the patterns are represented by a descriptor of 806 dimensions (i.e., all three descriptors are tested in this data set as described in [3]).
9. mAn [5]: this dataset contains protein data represented by 20 features and 20 labels.
10. Bibtex: this dataset is highly sparse and was used in [6].
11. Enron: a highly sparse dataset used in [6].
12. Health: a highly sparse dataset used in [6].

Table 1 shows a summary of the benchmarks along with their names, number of patterns, features, and labels, as well as the average number of class labels per pattern (LCard).

For dataset 6 (Liu), a 5-fold cross-validation testing protocol is used, and the results are averaged. For datasets 7 to 9, a 10-fold protocol is used. Datasets 1-5 and 10-12 are in the MATLAB IMCC toolkit [6] available at https://github.com/keauneuh/Incorporating-Multiple-Cluster-Centers-for-Multi-Label-Learning/tree/master/IMCCdata (accessed 9/24/22). All other datasets can be obtained from the references provided in their description above.

**Table 1.** Datasets summary.

| Name | #patterns | #features | #labels | LCard |
|---|---|---|---|---|
| CAL500 | 502 | 68 | 174 | 26.044 |
| Image | 2000 | 294 | 5 | 1.236 |
| Scene | 2407 | 294 | 5 | 1.074 |
| Yeast | 2417 | 103 | 14 | 4.24 |
| Arts | 5000 | 462 | 26 | 1.636 |
| ATC | 3883 | 42 | 14 | 1.265 |
| ATC_f | 3883 | 700 | 14 | 1.265 |
| Liu | 832 | 2892 | 1385 | 71.160 |
| mAn | 3916 | 20 | 20 | 1.650 |
| bibtex | 7395 | 1836 | 159 | 2.402 |
| enron | 1702 | 1001 | 53 | 3.378 |
| health | 5000 | 612 | 32 | 1.662 |

*2.2. Performance Indicators*

In the experimental section, several performance indicators are used to evaluate the classifiers on the multiclass benchmarks.

Let X be a dataset that includes m samples $\boldsymbol{x}_i \in \Re^d$. Each sample has an actual label $\boldsymbol{y}_i \in \{0,1\}^l$,, where l is the number of total labels. Let H and F be the set of predicted labels, where $\boldsymbol{h}_i \in \{0,1\}^l$ is the predicted label vector for sample $\boldsymbol{x}_i$, and $\boldsymbol{f}_i \in \Re^l$is the confidence relevance of each prediction. The performance indicators can be defined for H and F as follows:

- Hamming loss: this is the fraction of misclassified labels,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

$$\text{HLoss}(H) = \frac{1}{ml}\sum_{i=1}^{m}\sum_{j=1}^{l} I\left(\boldsymbol{y}_i(j) \neq \boldsymbol{h}_i(j)\right), \quad (1)$$

where I() is the indicator function. Hamming Loss should be minimized; if the hamming loss is 0, that means there is no error in the predicted label vector.

- One error: this is the fraction of instances whose most confident label is incorrect. Because this is an error, the indicator should be minimized:

$$\text{OneError}(F) = \frac{1}{m}\sum_{i=1}^{m} I\left(\boldsymbol{h}_i\left(\underset{j}{\text{argmax}}\,\boldsymbol{f}_i\right) \neq \boldsymbol{y}_i\left(\underset{j}{\text{argmax}}\,\boldsymbol{f}_i\right)\right), \quad (2)$$

- Ranking Loss: this is the average fraction of reversely ordered label pairs for each instance. It can be obtained from the confidence value considering the number of confidence couples correctly ranked (i.e., a true label is ranked before a wrong label). Ranking loss is also an error. Therefore, it should be minimized.
- Coverage: this is the average number of steps needed to move down the ranked label list of an instance to cover all its relevant labels. As such, coverage should be minimized.
- Average precision: this is the average fraction of relevant labels ranked higher than a particular label. As such, average precision should be maximized.

Another set of indicators adopted by many researchers [59] include:

- Aiming: this is the ratio of correctly predicted labels over the practically predicted labels:

$$\text{Aiming}(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{\|\boldsymbol{h}_i \cap \boldsymbol{y}_i\|}{\|\boldsymbol{h}_i\|}. \quad (3)$$

- Recall: this is the rate of the correctly predicted labels over the actual labels:

$$\text{Recall}(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{\|\boldsymbol{h}_i \cap \mathbf{y}_i\|}{\|\mathbf{y}_i\|}. \quad (4)$$

- Accuracy: this is the average ratio of correctly predicted labels over the total labels:

$$\text{Accuracy}\ (H) = \frac{1}{m}\sum_{i=1}^{m}\frac{\|\boldsymbol{h}_i \cap \mathbf{y}_i\|}{\|\boldsymbol{h} \cup \mathbf{y}_i\|}. \quad (5)$$

- Absolute true: this is the ratio of the perfectly correct prediction events over the total number of prediction events:

$$\text{AbsTrue}(H) = \frac{1}{m}\sum_{i=1}^{m} I(\boldsymbol{h}_i = \mathbf{y}_i). \quad (6)$$

- Absolute false: this is the ratio of the completely wrong prediction events over the total number of prediction events:

$$\text{AbsFalse}(H) = \frac{1}{m}\sum_{i=1}^{m}\frac{\|\boldsymbol{h}_i \cup \mathbf{y}_i\| - \|\boldsymbol{h}_i \cap \mathbf{y}_i\|}{l}. \quad (7)$$

The indicators listed above are within the range [0-1] and should be maximized, except for Absolute false.

## 3. Proposed Approach

### *3.1. Model Architecture*

As previously indicated, the Deep Neural Network (DNN) architectures developed in this work combine LSTM, GRU, and TCN networks that have been adapted to handle multilabel classification. The general schema for each model is available in Figure 1. GRU

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

with (N = 50) hidden units is followed by a max pooling and a fully connected layer. Multiclass classification is provided in the sigmoid output layer. TCN has a similar architecture, except that a fully connected layer is followed by a max pooling layer. These two architectures are labeled in this work GRU_A and TCN_A.

Experiments reveal that both GRU and TCN perform better in some situations when a convolutional level is placed immediately before the network itself. Convolution modifies input features with simple mathematical operations on other local features. These operations can produce better model generalization where features achieve higher special independence. When a convolutional level is attached before any TCN topologies, the network is labeled here as TCN_B.

In some GRU experiments, we add a batch-normalization layer immediately after a convolutional because batch normalization standardizes the inputs to a layer for each mini-batch, thereby stabilizing the learning process and dramatically reducing the number of training epochs required to train very deep networks. A GRU with a batch-normalization layer following a convolution is labeled GRU_B.

In addition, we investigate a sequential combination of GRU_A (without the pooling layer) followed by a TCN_A, where the sigmoid output of GRU_A becomes the input of TCN_A. This combination is labeled GRU_TCN.

The last architecture shown in Figure 1 is a network composed first of an LSTM layer with 125 hidden units followed by a dropout layer that randomly sets input elements to zero with a probability of 0.4. This is followed by a GRU layer with 100 hidden units and another dropout layer with a probability of 0.4. The end of the architecture is composed of a fully connected layer followed by a sigmoid layer.

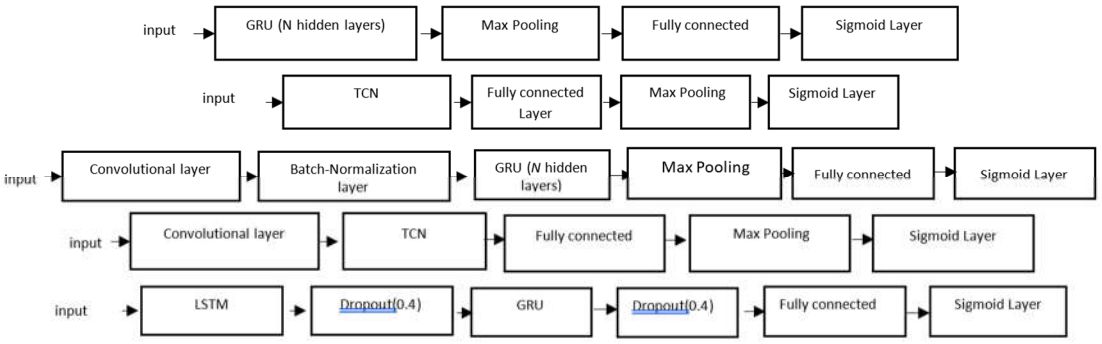



**Figure 1.** Schema of the proposed recurrent DNNs: GRU_A; TCN_A; GRU_B, TCN_B; LSTM_GRU.

The loss function is the binary cross entropy loss between the predicted labels (the output) and the actual labels (the target). Binary cross entropy loss calculates the loss of a set of m observations by computing the following average:

$$\text{CELoss} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{l}\mathbf{y}_i(j)\cdot\log\big(\mathbf{h}_i(j)\big) + (1-\mathbf{y}_i(j))\cdot\log\big(1-\mathbf{h}_i(j)\big). \quad (8)$$

where $\boldsymbol{y_i} \in \{0,1\}^l$ and $\boldsymbol{h_i} \in \{0,1\}^l$ are the actual and predicted label vectors of each sample $(i \in 1 \ldots m)$, respectively.

*3.2. Processing*

In the main, most dataset samples require no preprocessing before being fed into our proposed networks. However, some preprocessing is needed when feature vectors are very sparse.

Two types of preprocessing were applied in our experiments:

- Feature normalization in the range [0,1] for the dataset ATC_f for IMCC [6];

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

- For the datasets Liu, Arts, bibtex, enron, and health, feature transform was performed with PCA, where 99% of the variance was retained. Feature transform is only necessary for our proposed networks and not for IMCC and TB. Poor performance resulted when using the original sparse data as input to our proposed networks.

We also discovered that LSMT_GRU does not converge if a normalization step is not performed for the ATC_f dataset; it performs poorly even when the normalization step is performed. However, LSMT_GRU does converge when normalization is followed by PCA projection, where 99% of the variance is retained.

*3.3. Long short-term memory (LSTM)*

The LSTM [60] layer in our topologies learns long-term dependencies between the time steps in a time series and sequence data. This layer performs additive interactions, which can help improve gradient flow over long sequences during training.

LTSM can be defined as follows. Let the output or hidden state be $h_t$, and the cell state be $c_t$ at time step *t*. The first LSTM block uses the initial state of the network and the first time step of the sequence to compute the first output and updated cell state. At time step *t*, the block uses the current state of the network ($c_{t-1}$, $h_{t-1}$) and the next time step of the sequence to compute the output and the updated cell state ct.

The state of the layer is the hidden state/output state and the cell state. The hidden state at time step *t* contains the output of the LSTM layer for this time step. The cell state contains information learned from previous time steps. At each time step, the layer adds to or removes information from the cell state. The layer controls these updates using gates.

The basic components of a LSTM are an input gate, forget gate cell candidate, and output gate: the first determines the level of cell state update, the second the level of cell state reset (forget), the third adds information to cell state, and the fourth controls the level of the cell state added to the hidden state.

Given the above and letting $x_t$ be the input sequence with $h_0 = 0$, we can then define the input gate $i_t$, the forget gate $f_t$, the cell candidate $g_t$ and the output gate $o_t$ as:

$$i_t = o_g(W_i x_t + R_i h_{t-1} + b_i)$$

$$f_t = o_g(W_f x_t + R_f h_{t-1} + b_f)$$

$$g_t = o_c(W_g x_t + R_g h_{t-1} + b_g)$$

$$o_t = o_g(W_o x_t + R_o h_{t-1} + b_o)$$

where $W_i, R_i, b_i, W_f, R_f, b_f, W_g, R_g, b_g, W_o, R_o, b_o$ are matrices and vectors and $o_g$ denotes the gate activation function and σc the state activation function. The LSTM layer function, by default, uses the sigmoid function given by $\sigma(x) = (1 + e^{-x})^{-1}$ to compute the gate activation function.

We then define:

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t \tag{13}$$

as the cell state, where Θ is the Hadamard (component-wise) product.

The output vector is defined as:

$$h_t = o_t \odot \sigma_t(c_t) \tag{14}$$

The LSTM layer function, by default, uses the hyperbolic tangent function to compute the state activation function.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

### 3.4. Gated Recurrent Units (GRU)

GRU [41], like LTSM, is also a recurrent neural network with a gating mechanism. GRUs can handle the gradient vanishing problem and increase the length of term dependencies from the input. GRU has a forget gate that lets the network learn which old information is relevant for understanding the new information [61]. Unlike LSTM, GRU has fewer parameters because there is no output gate, yet the performance of GRU is similar to LSTM at many tasks: speech signal modeling, polyphonic music modeling, and natural language processing [8, 62]. They also perform better on small datasets [63] and work well on denoising tasks [64].

The basic components of GRU are a reset gate and an update gate, The Reset gate measures how much old information to forget. The reset gate decides which information to forget and which should be passed on to the output.

Letting $x_t$ be the input sequence and $h_0 = 0$, the update gate vector $z_t$ and the reset gate vector $r_t$ can be defined as

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{15}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \tag{16}$$

where $W_z, U_z, b_z, W_r, U_r$ and $b_r$ are matrices and vectors and σ is the sigmoid function.

We define

$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{17}$$

to be the candidate activation vector, where ϕ is the tanh activation, and Θ is the Hadamard (component-wise) product. The term $r_t$ determines the amount of past information for the candidate activation vector.

The output vector is

$$\hat{h}_t = \phi(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{18}$$

As can be observed, the update gate vector $z_t$ measures how much old and new information is combined and kept.

### 3.5. Temporal Convolutional Neural Networks (TCN)

TCNs [65] contain a hierarchy of one-dimensional convolutions stacked over time to form deep networks that perform fine-grained detection of events in sequence inputs. Subsequent layers are padded to match the size of the convolution stack, and the convolutions of each layer have a dilation factor that increases exponentially over the layers. This architecture allows the first layers to find short-term connections in the data while the deeper layers discover longer-term dependencies based on the features extracted by previous layers. Thus, TCNs have a large receptive field that bypasses a significant limitation of most RNN architectures.

The design of TCN blocks can vary considerably. The TCN block in this work is composed of a convolution of size three with 175 different filters, followed by a ReLU and batch normalization, followed by another convolution with the same parameters, a ReLU, and batch normalization. Four of these blocks are stacked. The dilation factors of the convolutions are $2^{k-1}$, with $k$ the number of a layer. On top, we use a fully connected layer, then a max pooling layer followed by the output layer, which is a sigmoid layer for multiclass classification. For training, we use dropout with a probability of 0.05.

### 3.6. IMCC

IMMC [6] is a two-step process that 1) creates virtual samples to augment the training set and 2) performs multilabel training. As augmentation is what provides IMCC its performance boost, the remainder of this discussion will be on the first step.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

Augmentation is performed with *k*-means clustering [66], along with the calculation of clustering centers. Let $\mathbf{X} = [x_1, x_2 \dots x_n]^T \in \mathbb{R}^{n \times d}$ be a feature matrix and $\mathbf{Y} = [y_1, y_2 \dots xy_n]^T \in (-1, +1)^{n \times q}$ be the label matrix, where $n$ is the number of samples. If all samples are partitioned into $c$ clusters $\{\mathcal{Z}_1, \mathcal{Z}_2 \dots \mathcal{Z}_c\}$ and $x_i$ is partitioned into cluster $\mathcal{Z}_j$ so that $x_i \in \mathcal{Z}_j$, then it can be assumed that the average of samples will capture the semantic meaning of the cluster.

The center $z_j$ of each cluster $\mathcal{Z}_i$ can be defined as

$$z_j = \frac{1}{|z_j|} \sum_{i=1}^{n} x \cdot \|( x_i \in \mathcal{Z}_j ), \tag{19}$$

where $\|(\cdot)$ is an indicator function that is equal to 1 when $x_i \in \mathcal{Z}_j$ or to 0, otherwise.

A complementary training set $\mathcal{D}' = \{z_j, t_j\}_{j=1}^{c}$ can be generated by averaging the label vectors of all instances of $\mathcal{Z}_i$, thus:

$$t_j = \frac{1}{|z_j|} \sum_{i=1}^{n} y_i \cdot \|( x_i \in \mathcal{Z}_j ). \tag{20}$$

For details on how the objective function handles the original data set $\mathcal{D}$ and the complementary dataset $\mathcal{D}'$, see [6]. In this study, the hyperparameters of IMCC are chosen by five-fold cross-validation on the training set.

### *3.7. Pooling*

Pooling layers (comprised of a single max along the time dimension) are added to the end of the GRU/TCN block to reduce the dimensionality of the processed data. In this way, only the most important information is retained, and the probability of overfitting is diminished.

### *3.8. Fully connected Layer and Sigmoid Layer*

The fully connected layer has *l* neurons, where *l* is the number of output labels in the given problem. This layer is fully connected with the previous layer. The activation function of this final layer is a sigmoid in the range [0…1]. These values are interpreted as the confidence relevance, or final probabilities, of each label. The output of the model is thus a multilabel classification vector, where the output of each neuron of the fully connected layer provides a score ranging from 0 to 1 for a single label in the set of labels.

### *3.9. Training*

As noted in the introduction, training is performed using different Adam variants as the optimizer. Each of these variants is discussed below in section 4. The learning rate is 0.01, and the gradient decay and squared gradient decay factors are 0.5 and 0.999, respectively.

In addition, gradients are cut off with a threshold of one using L2 norm gradient clipping. The minibatch size is fixed to 30, and the number of epochs in our experiments is set to 150 for GRU, LSTM_GRU, and GRU_TCN but 100 for TNC.

### *3.10. Ensemble Generation*

Ensembles combine the outputs of more than one model. In this work, models are trained on the same problem, and their decisions are fused using the average rule. The reason for constructing ensembles is that they improve system performance and prevent overfitting. It is well-known that an ensemble's prediction and generalization increase when the diversity among the classifiers is increased.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

A simple way of generating diversity in a set of neural networks is to initialize them randomly. However, applying different optimization strategies is a better way to strengthen diversity. Different optimization strategies can converge to different local minima and achieve different optima. In this work, we evaluate several Adam optimizers suitable for ensemble creation: the Adam optimizer [44], diffGrad [67] and four diffGrad variants: DGrad [68], Cos#1 [68], Exp [69], and Sto.

An ensemble of 40 networks is generated as follows: for each layer of each network, an optimization approach (DGrad, Cos#1, Exp, and Sto) is randomly selected for that layer.

## 4. Optimizers

### *4.1. Adam Optimizer*

Proposed in [44], Adam calculates the adaptive learning rates for each parameter by combining momentum and adaptive gradient. The Adam update rule is based on the value of the gradient at the current step and the exponential moving averages of the gradient and its square.

Specifically, the moving averages, $m_t$ (the first moment) and $u_t$ (the second moment), can be defined as

$$m_t = \rho_1 m_{t-1} + (1-\rho_1) g_t \tag{21}$$

$$u_t = \rho_2 u_{t-1} + (1-\rho_2) g_t^2 \tag{22}$$

where $g_t$ is the gradient at time $t$, the square on $g_t$ is the component-wise square, and $\rho_1$ and $\rho_2$ are hyperparameters representing the exponential decay rate for the first moment and the second moment estimates (usually set to 0.9 and 0.999), respectively, with moments initialized to 0: $m_o = u_0 = 0$.

To take into account the fact that the values of the moving averages are very small, at least in the first few steps, due to their initialization to zero, Adam is defined in such a way to correct for the bias of the moving averages:

$$\hat{m}_t = \frac{m_t}{(1-\rho_1^t)} \tag{23}$$

$$\hat{u}_t = \frac{u_t}{(1-\rho_2^t)} \tag{24}$$

The final update for each $\theta_t$ parameter of the network is

$$\theta_t = \theta_{t-1} - \lambda \frac{\hat{m}_t}{\sqrt{\hat{u}_t} + \epsilon} \quad , \tag{25}$$

where $\lambda$ is the learning rate, $\epsilon$ is a small positive number to prevent division by zero (usually set to $10^{-8}$), and all operations are component-wise.

Though $g_t$ can have positive or negative components, $g_t^2$ can have only positive components. It is possible, therefore, that in the case where the gradient changes sign often, the value of $\hat{m}_t$ could be much lower than $\sqrt{\hat{u}_t}$. In this case, the step size is very small.

### *4.2. The diffGrad optimizer*

The diffGrad optimizer, proposed in [67], uses the difference of the gradient to set the learning rate. Gradient changes begin to reduce during training, a behavior often indicating the presence of a global minima. The diffGrad optimizer takes advantage of this situation with an adaptive adjustment driven by the difference between the present and the immediate past values to lock parameters into a global minimum. This makes the step

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

size larger for faster gradient changes and smaller for lower gradient changes in parameters.

For diffGrad, the update is defined as the absolute difference of two consecutive steps of the gradient:

$$\Delta g_t = |g_{t-1} - g_t| \tag{26}$$

The final update for each $\theta_t$ parameter of the network is equation (28), where $\hat{m}_t$ and $\hat{u}_t$ are defined as in equations (23) and (24), but the learning rate is modulated by the Sigmoid of $\Delta g_t$ :

$$\xi_t = Sig(\Delta g_t) \tag{27}$$

$$\theta_{t+1} = \theta_t - \lambda \cdot \xi_t \frac{\hat{m}_t}{\sqrt{\hat{u}_t} + \epsilon} \tag{28}$$

*4.3. diffGrad Variants*

The following variants of the diffGrad optimization method are also used when building ensembles:

- DGrad [68] is based on the moving average of the element-wise squares of the parameter gradients;
- Cos#1 [68] is a minor variant of DGrad based on the application of a cyclic learning rate [70];
- Exp is a variant of a method proposed in [69] that is based on the application of an exponential function;
- Sto is a stochastic approach that stalls the optimizer on flat surfaces or small wells.

The proposed approaches have different methods for defining $\xi_t$, followed by the application of equation (28) to calculate the final update for $\theta_t$.

The rationale for all these variants is to avoid optimizer stalling on flat surfaces due to the low value of $\Delta g_t$.

**DGrad** [68] differs from diffGrad by defining the absolute difference between two consecutive steps of the gradient as

$$\Delta ag_t = |g_t - avg_t|, \tag{29}$$

where $avg_t$ contains the moving average of the element-wise squares of the parameter gradients; we then normalize $\Delta ag_t$ by its maximum as

$$\Delta \widehat{ag}_t = \left(\frac{\Delta ag_t}{\max(\Delta ag_t)}\right) \tag{30}$$

With DGrad, $\xi_t$ is defined as the Sigmoid of $4 \cdot \Delta \widehat{ag}_t$. The rationale for multiplying by four is to exploit the range of the output of the sigmoid function:

$$\xi_t = Sig(4 \cdot \Delta \widehat{ag}_t) \tag{31}$$

**Cos#1** [68] is a variant of DGrad with a cyclic learning rate. The idea is to improve classification accuracy without tuning and with fewer iterations [18].

In this work, the *cos*() periodic function is selected to define a range of variation in the learning rate:

$$lr_t = \left(2 - \left|\cos\left(\frac{\pi \cdot t}{steps}\right)\right| e^{-0.01 \cdot (mod(t, steps)+1)}\right), \tag{32}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

where *mod*() denotes the function modulo and where *steps*=30 is the period. The plot of $lr_t$ for $t$ in the range 1:2×*steps* is reported in Figure 1.

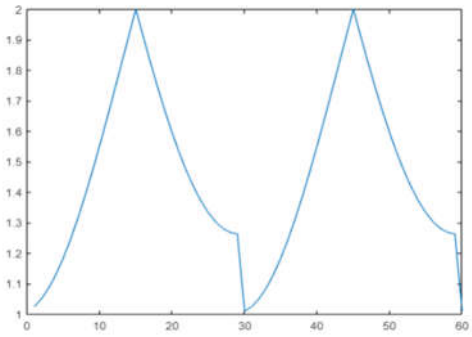



**Figure 1**. Cyclic learning rate.

With Cos#1, $lr_t$ is used as a multiplier of $\Delta\widehat{ag}_t$ in the definition of $\xi_t$ (equation (31):

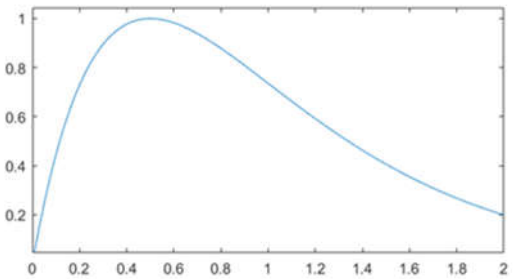
$$\xi_t = Sig(4 \cdot lr_t \cdot \Delta\widehat{ag}_t) \tag{33}$$

**Exp** [69] performs two simple element-wise operations: product and exponential. Exp is designed to mitigate the effect of large variations in the gradient and help the function converge for small values. The function in eq. (34) has a pattern that decays slower than the negative exponential for larger values; moreover, thanks to normalization, it gives less focus on gradient variations that tend to zero, widening the area of greater gain, thus:

$$lr_t = \Delta ag_t \cdot e^{(-k \cdot \Delta ag_t)}, \tag{34}$$

with $k$ fixed at 2 in this work.

The final learning rate $(\xi_t)$ is the normalized result multiplied by a correction factor (1.5), which helps to move the mean towards the unit (see the plot reported in **Error! Reference source not found.**):




$$\xi_t = 1.5 \frac{lr_t}{\max(lr_t)} \tag{35}$$

**Figure 3**. Plot of equation (35).

**Sto** (short for Stochastic), proposed here, adds noise to the learning rate to reduce the possibility that the optimizer will stall on flat surfaces or small wells. The noise is independent of the gradient direction.

Let $\mathcal{X}$ be a matrix of independent uniform random variables in range [0,1] and $J$ an all-ones matrix:

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

$$\mathcal{X} = \begin{bmatrix} X_{1,1} & \cdots & X_{1,n} \\ \vdots & \ddots & \vdots \\ X_{m,1} & \cdots & X_{m,n} \end{bmatrix} \text{ and } J = \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{bmatrix},$$

where $X_{i,j} \sim \mathcal{U}(0,1)$ are random variables with a uniform density function. The learning rate is defined as

$$lr_t = \Delta ag_t \cdot e^{(-4 \cdot \Delta ag_t)} \cdot (\mathcal{X} + 0.5 \cdot J) \tag{36}$$

$$\xi_t = 1.5 \frac{lr_t}{\max(lr_t)} \tag{37}$$

In eq. 36, matrix $J$ shifts the range of $X$ by 0.5 to move the mean across 1.

## 5. Experimental Results

The goal of the first experiment was to build and evaluate the performance of the different variants of the base models combined with all the components detailed in sections 3 and 4. All ensembles were fused by the average rule. In Table 2, we provide a descriptive summary of each of these ensembles: the number of classifiers and hidden units, the number of training epochs, and the loss function. Starting from a standalone GRU_A with 50 hidden units trained by Adam for 50 epochs (labeled Adam_sa), we generated new ensembles by incrementally increasing the complexity of the base GRU by fusing 10 Adam_sa (Adam_10s), increasing the number of epochs to 150 (Adam_10), changing the optimizer (DG_10, Cos_10, Exp_10, Sto_10), and fusing the best results as follows:

- DG_Cos is the fusion of DG_10 + Cos_10;
- DG_Cos_Exp is the fusion of DG_10 + Cos_10 + Exp_10;
- DG_Cos_Exp_Sto is the fusion of DG_10 + Cos_10 + Exp_10 + Sto_10;
- StoGRU is an ensemble composed of 40 GRU_A, combined by average rule, each coupled with the stocastic approach explained in section 3.10;
- StoGRU_B as StoGRU but based on GRU_B;
- StoTCN is an ensemble of 40 TCN_A, combined by average rule, each coupled with the stocastic approach explained in section 3.10;
- StoTCN_B as StoTCN but based on TCN_B;
- StoGRU_TCN is an ensemble of 40 GRU_TCN each coupled with the stocastic approach explained in section 3.10;
- StoLSTM_GRU is an ensemble of 40 LSTM_GRU each coupled with the stocastic approach explained in section 3.10;
- ENNbase is the fusion by average rule of StoGRU and StoTCN;
- ENN is the fusion by average rule of StoGRU, StoTCN, StoGRU_B, StoTCN_B and StoGRU_TCN;
- ENNlarge is the fusion by average rule of StoGRU, StoTCN, StoGRU_B, StoTCN_B, StoGRU_TCN and StoLSTM_GRU.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

**Table 2.** Summary of tested ensembles.

| Name | Hidden units | #Classifiers | #Epoch | Optimizer |
|---|---|---|---|---|
| Adam_sa | 50 | 1 | 50 | Adam |
| Adam_10s | 50 | 10 | 50 | Adam |
| Adam_10 | 50 | 10 | 150 | Adam |
| DG_10 | 50 | 10 | 150 | DGrad |
| Cos_10 | 50 | 10 | 150 | Cos |
| Exp_10 | 50 | 10 | 150 | Exp |
| Sto_10 | 50 | 10 | 150 | Sto |
| DG_Cos= DG_10 + Cos_10 | 50 | 20 | 150 | DGrad,Cos |
| DG_Cos_Exp = DG_10 + Cos_10 + Exp_10 | 50 | 30 | 150 | DGrad,Cos, Exp |
| DG_Cos_Exp_Sto = DG_10 + Cos_10 + Exp_10 + Sto_10 | 50 | 40 | 150 | DGrad,Cos, Exp, Sto |
| StoGRU | 50 | 40 | 150 | DGrad,Cos, Exp, Sto |
| StoGRU_B | 50 | 40 | 150 | DGrad,Cos, Exp, Sto |
| StoTCN | --- | 40 | 100 | DGrad,Cos, Exp, Sto |
| StoTCN_B | --- | 40 | 100 | DGrad,Cos, Exp, Sto |
| StoGRU_TCN | 50 (GRU) | 40 | 150 | DGrad,Cos, Exp, Sto |
| StoLSTM_GRU | 125 (LSTM layer) 100 (GRU layer) | 40 | 150 | DGrad,Cos, Exp, Sto |
| ENNbase=StoTCN+StoGRU | 50 (GRU) | 80 | 100 (TCN) / 150 (GRU) | DGrad,Cos, Exp, Sto |
| ENN=StoTCN+StoGRU+StoTCN_B+StoGRU_B+StoGRU_TCN | 50 (GRU, GRU_B & GRU_TCN ) | 200 | 100 (TCN & TCN_B) / 150 (GRU, GRU_B & GRU_TCN) | DGrad,Cos, Exp, Sto |
| ENNlarge=StoTCN+StoGRU+StoTCN_B+StoGRU_B+StoGRU_TCN+StoLSTM_GRU | 50 (GRU, GRU_B & GRU_TCN ) 100/125 (LSTM_GRU) | 240 | 100 (TCN & TCN_B) / 150 (GRU, GRU_B, LSTM_GRU & GRU_TCN) | DGrad,Cos, Exp, Sto |

An ablation study for assessing the performance improvement that each module of our ensemble achieved is reported in Table 3. Only tests on GRU_A is reported here. The others topologies tested in this work produced similar conclusions. In table 3, we compare approaches using Wilcoxon signed rank test.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

**Table 3.** Ablation study showing that StoGRU is the best method among the approaches based on GRU_A (other approaches produced similar results).

| Method | Comparison |
|---|---|
| Adam_10s | Outperforms Adam_sa with a p-value of 0.0156 |
| Adam_10 | Outperforms Adam_sa with a p-value of 0.0172<br>Same performance of Adam_10s |
| DG_10 | Outperforms Adam_10 with a p-value of 0.0064 |
| Cos_10 | Outperforms Adam_10 with a p-value of 0.0137 |
| Exp_10 | Outperforms Adam_10 with a p-value of 0.0016 |
| Sto_10 | Outperforms Adam_10 with a p-value of 0.1014 |
| DG_Cos_Exp_Sto | Outperforms Exp_10 (the best of the approaches reported above in this table) with a p-value of 0.0134 |
| StoGRU | Outperforms DG_Cos_Exp_Sto with a p-value of 0.0922 |

Table 4 shows the performance of the ensembles listed in Table 2 in terms of average precision. Moreover, we report the performance of IMCC [6] and TB. For both IMCC and TB, the hyper-parameters were chosen by a five-fold cross-validation on the training set. We also report different fusions among ENNlarge, IMCC, and TB:

- ENNlarge+$w$×IMCC is the sum rule between ENNlarge and IMCC; before fusion, the scores of ENNlarge (notice that the ensemble ENNlarge is obtained by average rule) were normalized since it has a different range of values compared to IMCC. Normalization was performed as $ENNlarge = (ENNlarge - 0.5) \times 2$, the classification threshold of the the ensemble is simply set to zero. The scores of IMCC were weighted by a factor of $w$.
- ENNlarge+$w$×IMCC+TB is the same as the previous fusion, but TB is included in the ensemble. Before fusion, the scores of TB were normalized since it has a different range of values compared to IMCC. Normalization was performed as $TB = (TB - 0.5) \times 2$.
- StoLSTM_GRU+IMCC+TB is the sum rule among StoLSTM_GRU, IMCC, and TB. StoLSTM_GRU and TB are normalized before the fusion as $StoLSTM\_GRU = (StoLSTM\_GRU - 0.5) \times 2$; $TB = (TB - 0.5) \times 2$.

IMCC and TB do not use PCA when sparse datasets are tested, as noted in section 3.2.

**Table 4.** Average precision of the ensembles and state of the art on the twelve benchmarks (boldface values indicate the best performance within each group of similar approaches).

| Average Precision | Cal500 | image | scene | yeast | arts | ATC | ATC_f | Liu | mAn | bibtex | enron | health | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| IMCC | **0.502** | 0.836 | 0.904 | 0.773 | 0.619 | 0.866 | **0.922** | 0.523 | 0.978 | **0.623** | **0.714** | 0.781 | 0.753 |
| TB | 0.489 | 0.844 | 0.873 | **0.778** | 0.625 | 0.882 | 0.897 | 0.518 | **0.983** | 0.572 | 0.701 | 0.753 | 0.743 |
| StoGRU | 0.498 | 0.851 | 0.911 | 0.740 | 0.561 | 0.872 | 0.872 | 0.485 | 0.979 | 0.403 | 0.680 | 0.739 | 0.715 |
| StoGRU_B | 0.490 | **0.861** | 0.908 | 0.741 | 0.555 | 0.877 | 0.848 | 0.485 | 0.978 | 0.400 | 0.688 | 0.724 | 0.712 |
| StoTCN | 0.498 | 0.847 | 0.913 | 0.764 | 0.506 | 0.882 | 0.900 | 0.498 | 0.977 | 0.406 | 0.669 | 0.710 | 0.714 |
| StoTCN_B | 0.497 | 0.855 | **0.917** | 0.765 | 0.541 | 0.883 | 0.903 | 0.505 | 0.976 | 0.404 | 0.666 | 0.732 | 0.720 |
| StoGRU_TCN | 0.491 | 0.852 | 0.916 | 0.752 | 0.592 | **0.890** | 0.913 | 0.510 | 0.977 | 0.354 | 0.674 | 0.764 | 0.724 |
| StoLSTM_GRU | 0.493 | 0.839 | 0.901 | 0.771 | **0.633** | 0.888 | 0.912 | **0.541** | 0.978 | 0.618 | 0.702 | **0.790** | **0.756** |
| ENNbase | 0.502 | 0.855 | 0.922 | 0.756 | 0.552 | 0.888 | 0.916 | 0.497 | 0.979 | 0.417 | 0.687 | 0.735 | 0.726 |
| ENN | 0.499 | 0.859 | **0.924** | 0.762 | 0.582 | **0.893** | 0.916 | 0.505 | 0.979 | 0.424 | 0.689 | 0.749 | 0.732 |
| ENNlarge | 0.498 | **0.860** | 0.923 | 0.776 | 0.628 | 0.892 | 0.926 | 0.520 | 0.979 | 0.534 | 0.708 | 0.780 | 0.752 |
| ENNlarge+IMCC | 0.502 | 0.853 | 0.920 | 0.784 | 0.633 | 0.883 | 0.926 | 0.526 | 0.979 | 0.627 | 0.717 | 0.790 | 0.762 |
| ENNlarge+3×IMCC | **0.503** | 0.847 | 0.913 | 0.778 | 0.628 | 0.875 | 0.925 | 0.526 | 0.979 | 0.626 | 0.718 | 0.787 | 0.759 |
| ENNlarge+IMCC+TB | 0.500 | 0.856 | 0.913 | 0.784 | **0.641** | 0.889 | **0.927** | 0.526 | 0.982 | 0.622 | 0.718 | 0.788 | 0.762 |
| ENNlarge+3×IMCC+TB | 0.502 | 0.850 | 0.910 | 0.783 | 0.637 | 0.880 | 0.926 | 0.527 | 0.981 | 0.627 | 0.717 | 0.787 | 0.761 |
| StoLSTM_GRU+IMCC+TB | 0.500 | 0.851 | 0.898 | **0.786** | **0.641** | 0.883 | 0.920 | **0.538** | **0.983** | **0.635** | **0.727** | **0.800** | **0.764** |

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

These are some of the conclusions that can be drawn from examining Table 4:

- GRU/TCN-based methods work poorly on very sparse datasets (i.e., on Arts, Liu, bibtex, enron, and health);
- StoGRU_TCN outperforms the other ensembles based on GRU/TCN; StoGRU and StoTCN perform similarly;
- StoLSTM_GRU works very well on sparse datasets. On datasets that are not sparse, the performance is similar to that obtained by the other methods based on GRU/TCN. StoLSTM_GRU average performance is higher than that obtained by IMCC;
- ENNlarge outperforms each method from which it was built;
- ENNlarge+3×IMCC+TB outperforms ENNlarge+3×IMCC with a p-value 0.09;
- StoLSTM_GRU+IMCC+TB is the best choice for sparse datasets;
- ENNlarge+3×IMCC+TB tops or equals IMCC in all the datasets (note that ENN+IMCC+TB and StoLSTM_GRU+IMCC+TB have performance equal to or lower than IMCC in some datasets). ENNlarge+3×IMCC+TB is our suggested approach.

In the following tests, we simplify the names of our best ensembles to reduce clutter:

- Ens refers to ENNlarge+3×IMCC+TB;
- EnsSparse refers to StoLSTM_GRU+IMCC+TB.

In Table 5, we compare IMCC and Ens using more performance indicators. Our ensemble Ens outperforms IMCC.

**Table 5.** Comparison of IMCC and our proposed ensemble ENS using five performance indicators.

| | One Error ↓ | Hamming Loss ↓ | Ranking Loss ↓ | Coverage ↓ | Avg Precision ↑ |
|---|---|---|---|---|---|
| Cal500-IMCC | **0.150** | 0.134 | 0.182 | 0.736 | 0.502 |
| Cal500-Ens | **0.150** | **0.134** | **0.179** | **0.729** | **0.502** |
| Image-IMCC | 0.237 | 0.150 | 0.138 | 0.173 | 0.836 |
| Image-Ens | **0.225** | **0.147** | **0.127** | **0.159** | **0.850** |
| scene-IMCC | 0.164 | 0.070 | 0.053 | 0.062 | 0.904 |
| scene-Ens | **0.151** | **0.067** | **0.047** | **0.057** | **0.910** |
| Yest-IMCC | 0.220 | 0.185 | 0.165 | 0.448 | 0.773 |
| Yest-Ens | **0.211** | **0.178** | **0.155** | **0.433** | **0.783** |
| Arts-IMCC | 0.438 | 0.054 | 0.164 | 0.242 | 0.619 |
| Arts-Ens | **0.431** | **0.053** | **0.144** | **0.219** | **0.637** |
| Bibtex-IMCC | **0.336** | **0.012** | 0.079 | 0.158 | 0.623 |
| Bibtex-Ens | 0.338 | **0.012** | **0.072** | **0.143** | **0.627** |
| Enron-IMCC | 0.226 | **0.044** | 0.072 | 0.211 | 0.714 |
| Enron-Ens | **0.226** | **0.044** | **0.069** | **0.204** | **0.717** |
| Health-IMCC | 0.266 | **0.035** | 0.052 | 0.107 | 0.781 |
| Health-Ens | **0.262** | **0.035** | **0.046** | **0.097** | **0.787** |

In table 6, we compare Ens with state of the art on the mAn dataset using the performance measures covered in the literature for this dataset, *viz*., coverage, accuracy, absolute true, and absolute false.

**Table 6.** Performance comparison of Ens and IMCC on the mAn data set.

| mAn | Aiming | Coverage | Accuracy | Absolute True | Absolute False |
|---|---|---|---|---|---|
| [71] | 88.31 | 85.06 | 84.34 | 78.78 | 0.07 |
| [5] | **96.21** | **97.77** | **95.46** | **92.26** | **0.00** |
| IMCC | 92.80 | 92.02 | 88.83 | 80.76 | 1.43 |
| Ens | 93.84 | 93.06 | 90.24 | 83.64 | 1.20 |

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 17 October 2022 doi:10.20944/preprints202210.0224.v1

Ens outperforms IMCC on mAn on these five measures; however, Ens does not achieve state-of-the-art performance. An ad hoc method recently proposed in [5] performs better on this dataset than Ens.

In table 7, we show the performance reported in a recent survey [55] of many multilabel methods (the meaning of the acronyms and sources are noted in [55] and not included here as the intention of this table to demonstrate the superiority of Ens compared to the score reported in the survey). The last column reports the performance of our proposed ensemble. As can be observed, it obtains the average best performance.

**Table 7.** Comparisons with results reported in a recent survey [55] using average precision as the performance indicator.

| | EPS | CDE | MLkNN | MLARM | BR | DEEP1 | PCT | HOMER | AdaBoost.MH | BPNN | RAkEL | CLR | RFPCT | PSt | TREMLC | RFDTBR | MBR | CDN | ECCJ48 | EBRJ48 | DEEP4 | RSLP | CLEMS | Ens |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| bibtex | 0.466 | 0.414 | 0.161 | 0.423 | 0.350 | 0.335 | 0.016 | 0.316 | 0.472 | 0.434 | 0.081 | 0.463 | 0.515 | 0.538 | 0.483 | 0.559 | 0.265 | 0.231 | 0.492 | 0.546 | 0.258 | 0.491 | 0.183 | **0.627** |
| Cal500 | 0.440 | 0.411 | 0.441 | 0.352 | 0.236 | 0.489 | 0.497 | 0.288 | 0.475 | 0.508 | 0.271 | 0.355 | **0.520** | 0.485 | 0.497 | 0.500 | 0.381 | 0.292 | 0.427 | 0.458 | 0.329 | 0.490 | 0.446 | 0.502 |
| enron | 0.622 | 0.580 | 0.517 | 0.529 | 0.512 | 0.624 | 0.531 | 0.388 | 0.627 | 0.642 | 0.447 | 0.623 | 0.683 | 0.629 | 0.675 | 0.686 | 0.498 | 0.485 | 0.623 | 0.675 | 0.537 | 0.548 | 0.538 | **0.717** |
| scene | 0.789 | 0.812 | 0.785 | 0.715 | 0.790 | 0.686 | 0.745 | 0.753 | 0.880 | 0.855 | 0.851 | 0.889 | 0.868 | 0.887 | 0.856 | 0.874 | 0.711 | 0.501 | 0.813 | 0.856 | 0.810 | 0.797 | 0.850 | **0.910** |
| yeast | 0.745 | 0.718 | 0.704 | 0.598 | 0.663 | 0.701 | 0.732 | 0.693 | 0.711 | 0.761 | 0.693 | 0.710 | 0.762 | 0.766 | 0.760 | 0.763 | 0.576 | 0.437 | 0.719 | 0.740 | 0.709 | 0.748 | 0.715 | **0.787** |

Finally, in Table 8, our best ensembles, Ens and EnsSparse, are compared with the literature across all twelve benchmarks using average precision as the performance indicator. Ens and EnsSparse obtain state-of-the-art performance using this measure on several of the multilabel benchmarks.

**Table 8.** Other comparisons with the literature using average precision.

| Average Precision | Cal500 | image | scene | yeast | arts | ATC | ATC_f | Liu | mAn | bibtex | enron | health |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Ens | 0.503 | 0.849 | **0.912** | 0.780 | 0.632 | 0.878 | **0.926** | 0.527 | 0.981 | 0.626 | 0.717 | 0.788 |
| EnsSparse | 0.500 | **0.851** | 0.898 | 0.786 | **0.641** | **0.883** | 0.920 | **0.538** | **0.983** | **0.635** | 0.727 | **0.800** |
| FastAi [33] | 0.425 | 0.824 | 0.899 | 0.718 | 0.588 | 0.860 | 0.908 | 0.414 | 0.976 | --- | --- | --- |
| IMCC | 0.502 | 0.836 | 0.904 | 0.773 | 0.619 | 0.866 | 0.922 | 0.523 | 0.978 | 0.623 | 0.714 | 0.781 |
| hML | 0.453 | 0.810 | 0.885 | **0.792** | 0.538 | 0.831 | 0.854 | 0.433 | 0.965 | --- | --- | --- |
| ECC [6] | 0.491 | 0.797 | 0.857 | 0.756 | 0.617 | --- | --- | --- | --- | 0.617 | 0.657 | 0.719 |
| MAHR [6] | 0.441 | 0.801 | 0.861 | 0.745 | 0.524 | --- | --- | --- | --- | 0.524 | 0.641 | 0.715 |
| LLSF [6] | 0.501 | 0.789 | 0.847 | 0.617 | 0.627 | --- | --- | --- | --- | 0.627 | 0.703 | 0.780 |
| JFSC [6] | 0.501 | 0.789 | 0.836 | 0.762 | 0.597 | --- | --- | --- | --- | 0.597 | 0.643 | 0.751 |
| LIFT [6] | 0.496 | 0.789 | 0.859 | 0.766 | 0.627 | --- | --- | --- | --- | 0.627 | 0.684 | 0.708 |
| [16] | --- | --- | --- | --- | --- | --- | --- | 0.513 | --- | --- | --- | --- |
| [72] | --- | --- | --- | --- | --- | --- | --- | 0.261 | --- | --- | --- | --- |
| hMuLab [19] | --- | --- | --- | 0.778 | --- | --- | --- | --- | --- | --- | --- | --- |
| MlKnn [19] | --- | --- | --- | 0.766 | --- | --- | --- | --- | --- | --- | --- | --- |
| RaKel [19] | --- | --- | --- | 0.715 | --- | --- | --- | --- | --- | --- | --- | --- |
| ClassifierChain [19] | --- | --- | --- | 0.624 | --- | --- | --- | --- | --- | --- | --- | --- |
| IBLR [19] | --- | --- | --- | 0.768 | --- | --- | --- | --- | --- | --- | --- | --- |
| MLDF [73] | 0.512 | 0.842 | 0.891 | 0.770 | --- | --- | --- | --- | --- | --- | **0.742** | --- |
| RF_PCT [73] | 0.512 | 0.829 | 0.873 | 0.758 | --- | --- | --- | --- | --- | --- | 0.729 | --- |
| DBPNN [73] | 0.495 | 0.672 | 0.563 | 0.738 | --- | --- | --- | --- | --- | --- | 0.679 | --- |
| MLFE [73] | 0.488 | 0.817 | 0.882 | 0.759 | --- | --- | --- | --- | --- | --- | 0.656 | --- |
| ECC [73] | 0.482 | 0.739 | 0.853 | 0.752 | --- | --- | --- | --- | --- | --- | 0.646 | --- |
| RAKEL [73] | 0.353 | 0.788 | 0.843 | 0.720 | --- | --- | --- | --- | --- | --- | 0.596 | --- |
| [15] | --- | --- | --- | 0.758 | --- | --- | --- | --- | --- | --- | --- | --- |
| [74] | 0.484 | --- | --- | 0.740 | --- | --- | --- | --- | --- | --- | --- | --- |
| [75] | --- | --- | --- | 0.775 | 0.636 | --- | --- | --- | --- | --- | --- | --- |
| Wrap [76] | **0.520** | --- | 0.832 | 0.761 | --- | --- | --- | --- | --- | 0.578 | 0.710 | --- |
| Wrap$^k$ [76] | 0.518 | --- | 0.892 | 0.781 | --- | --- | --- | --- | --- | 0.571 | 0.720 | --- |

 doi:10.20944/preprints202210.0224.v1

## 6. Conclusion

The system proposed in this work for multilabel classification is composed of ensembles of gated recurrent units (GRU), temporal convolutional neural networks (TCN) and long short-term memory networks (LSTM) trained with several variants of Adam optimization. This approach is combined with Incorporating Multiple Clustering Centers (IMCC) to produce a superior multilabel classification system. Many ensembles are tested across a set of twelve multilabel benchmarks representing many different applications. Experimental results show that the best ensemble constructed using our novel approach obtains state-of-the-art performance.

Future studies will focus on the fusion of this approach with other topologies for extracting features. Moreover, more sparse datasets will be used to evaluate the performance of the proposed ensemble for further validation of the conclusions drawn in this work.

**Author Contributions:** For research articles with several authors, a short paragraph specifying their individual contributions must be provided. The following statements should be used "Conceptualization, L.N. and L.T.; methodology, L.N.; validation, X.X., Y.Y. and Z.Z.; formal analysis, L.N, S.B, X.G, and C.W.; writing—review and editing, L.N, S.B, X.G, and C.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding

**Data Availability Statement:** https://github.com/LorisNanni

**Acknowledgments:** Through their GPU Grant Program, NVIDIA donated the TitanX GPU that was used to train the CNNs presented in this work.

**Conflicts of Interest:** The authors declare no conflict of interest

## References

1. E. G. Galindo and S. Ventura, "Multi label learning: a review of the state of the art and ongoing research," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery,* vol. 4, no. 6, pp. 411-444, 2014, doi: doi.org/10.1002/widm.1139.
2. A. Elisseeff and J. Weston, *A kernel method for multi-labelled classification* (NIPS). MIT Press Direct, 2001.
3. L. Nanni, A. Lumini, and S. Brahnam, "Neural networks for anatomical therapeutic chemical (ATC) classification," *Applied Computing and Informatics,* 2022.
4. L. Chen *et al.*, "Predicting gene phenotype by multi-label multi-class model based on essential functional features," *Molecular genetics and genomics : MGG,* vol. 296, no. 4, pp. 905-918, 2021, doi: 10.1007/s00438-021-01789-8.
5. Y. Shao and K. Chou, "pLoc_Deep-mAnimal: A Novel Deep CNN-BLSTM Network to Predict Subcellular Localization of Animal Proteins," *Natural Science,* vol. 12, 5, pp. 281-291, 2020, doi: 10.4236/ns.2020.125024.
6. S. Shu *et al.*, "Incorporating Multiple Cluster Centers for Multi-Label Learning," *ArXiv,* vol. abs/2004.08113, 2020.
7. M. Ibrahim, M. U. G. Khan, F. Mehmood, M. Asim, and W. Mahmood, "GHS-NET a generic hybridized shallow neural network for multi-label biomedical text classification," *Journal of biomedical informatics,* vol. 116, p. 103699, 2021, doi: 10.1016/j.jbi.2021.103699.
8. M. Ravanelli, P. Brakel, M. Omologo, and Y. Bengio, "Light Gated Recurrent Units for Speech Recognition," *IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 2, no. 2, pp. 92-102, 2018, doi: 10.1109/TETCI.2017.2762739.
9. Y. Kim and J. Kim, "Human-Like Emotion Recognition: Multi-Label Learning from Noisy Labeled Audio-Visual Expressive Speech," presented at the 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2018.
10. M. B. Messaoud, I. Jenhani, N. B. Jemaa, and M. W. Mkaouer, "A Multi-label Active Learning Approach for Mobile App User Review Classification," in *KSEM,* 2019.
11. J. P. Singh and K. Nongmeikapam, "Negative Comments Multi-Label Classification," *2020 International Conference on Computational Performance Evaluation (ComPE),* pp. 379-385, 2020, doi: 10.1109/ComPE49325.2020.9200131.
12. M. Boutell, J. Luo, X. Shen, and C. M. Brown, "Learning multi-label scene classification," *Pattern Recognit.,* vol. 37, no. 9, pp. 1757-1771, 2004.
13. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining Multi-label Data," in *Data Mining and Knowledge Discovery Handbook,* 2010: Springer, pp. 667–685.
14. J. Read, B. Pfahringer, G. Holmes, and E. Frank, "Classifier chains for multi-label classification," *Mach Learn,* vol. 85, pp. 333-359, 2011.
15. W. Qian, C. Xiong, and Y. Wang, "A ranking-based feature selection for multi-label classification with fuzzy relative discernibility," *Applied Soft Computing,* vol. 102, p. 106995, 2021/04/01/ 2021, doi: https://doi.org/10.1016/j.asoc.2020.106995.

16. W. Zhang, F. Liu, L. Luo, and J. Zhang, "Predicting drug side effects by multi-label learning and ensemble learning," *BMC Bioinformatics,* vol. 16, no. 365, 2015.
17. J. Huang, G. Li, Q. Huang, and X. Wu, "Joint Feature Selection and Classification for Multilabel Learning," *IEEE Transactions on Cybernetics,* vol. 48, no. 3, pp. 876-889, 2018.
18. J. Huang, G. Li, Q. Huang, and X. Wu, "Learning Label-Specific Features and Class-Dependent Labels for Multi-Label Classification," *IEEE Transactions on Knowledge and Data Engineering,* vol. 28, no. 12, pp. 3309-3323, 2016.
19. P. Wang, R. Ge, X. Xiao, M. Zhou, and F. Zhou, "hMuLab: A Biomedical Hybrid MUlti-LABel Classifier Based on Multiple Linear Regression," *IEEE/ACM Trans. Comput. Biol. Bioinformatics,* vol. 14, no. 5, pp. 1173–1180, 2017, doi: 10.1109/tcbb.2016.2603507.
20. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Random k-Labelsets for Multilabel Classification," *IEEE Transactions on Knowledge and Data Engineering,* vol. 23, no. 7, pp. 1079-1089, 2011.
21. Y. Yang and J. Jiang, "Adaptive Bi-Weighting Toward Automatic Initialization and Model Selection for HMM-Based Hybrid Meta-Clustering Ensembles," *IEEE Transactions on Cybernetics,* vol. 49, no. 5, pp. 1657-1668, 2019.
22. J. M. Moyano, E. G. Galindo, K. Cios, and S. Ventura, "Review of ensembles of multi-label classifiers: Models, experimental study and prospects," *Inf. Fusion,* vol. 44, no. November, pp. 33-45, 2018.
23. Y. Xia, K. Chen, and Y. Yang, "Multi-label classification with weighted classifier selection and stacked ensemble," *Inf. Sci.,* vol. 557, no. May, pp. 421-442, 2021.
24. J. M. Moyano, E. G. Galindo, K. Cios, and S. Ventura, "An evolutionary approach to build ensembles of multi-label classifiers," *Inf. Fusion,* vol. 50, no. October, pp. 168-180, 2019.
25. R. Wang, S. Kwong, X. Wang, and Y. Jia, "Active k-labelsets ensemble for multi-label classification," *Pattern Recognit.,* vol. 109, no. January, p. 107583, 2021.
26. DeepDetect. "DeepDetect." https://www.deepdetect.com/ (accessed 2021).
27. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," Cornell University, arXiv:1409.1556v6 2014.
28. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," presented at the IEEE Conference on Computer Vision and Pattern Recognition, 2016.
29. C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning," in "arxiv.org," Cornell University, https://arxiv.org/pdf/1602.07261.pdf, 2016.
30. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* pp. 770-778, 2015.
31. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 18-23 June 2018 2018, pp. 4510-4520, doi: 10.1109/CVPR.2018.00474.
32. J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *ArXiv,* vol. abs/1804.02767, 2018.
33. J. Howard and S. Gugger, "Fastai: A Layered API for Deep Learning," *Information,* vol. 11, no. 2, p. 108, 2020. [Online]. Available: https://www.mdpi.com/2078-2489/11/2/108.
34. Imagga. "Imagga website." https://imagga.com/solutions/auto-tagging (accessed 2021).
35. Wolfram. "Wolfram Alpha: Image Identification Project." https://www.imageidentify.com/ (accessed 2020).
36. Clarifai. "Clarifai website." https://www.clarifai.com/ (accessed 2021).
37. Microsoft. "Computer-vision API website." https://www.microsoft.com/cognitive-services/en-us/computer-vision-api (accessed 2021).
38. IBM. "Visual Recognition." https://www.ibm.com/watson/services/visual-recognition/ (accessed 2020).
39. Google. "Google Cloud Vision." https://cloud.google.com/vision/ (accessed 2021).
40. A. Kubany, S. B. Ishay, R.-s. Ohayon, A. Shmilovici, L. Rokach, and T. Doitshman, "Comparison of state-of-the-art deep learning APIs for image multi-label classification using semantic metrics," *Expert Syst. Appl.,* vol. 161, no. 15, p. 113656, 2020.
41. K. Cho *et al.*, "Learning Phrase Representations using RNN Encoder Decoder for Statistical Machine Translation," in *EMNLP*, Varna, Bulgaria, 2014, pp. 25-32.
42. C. S. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. Hager, "Temporal Convolutional Networks for Action Segmentation and Detection," presented at the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii, 2017.
43. L. Nanni, A. Lumini, A. Manfe, S. Brahnam, and G. Venturin, "Gated recurrent units and temporal convolutional network for multilabel classification," *arXiv preprint arXiv:2110.04414,* 2021.
44. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *CoRR,* vol. abs/1412.6980, 2015.
45. N. Meinshausen and G. Ridgeway, "Quantile regression forests," *Journal of machine learning research,* vol. 7, no. 6, 2006.
46. M.-L. Zhang and Z.-H. Zhou, "Multilabel Neural Networks with Applications to Functional Genomics and Text Categorization," *IEEE Transactions on Knowledge and Data Engineering,* vol. 18, no. 10, pp. 1338-1351, 2006, doi: 10.1109/TKDE.2006.162.
47. R. Stivaktakis, G. Tsagkatakis, and P. Tsakalides, "Deep Learning for Multilabel Land Cover Scene Categorization Using Data Augmentation," *IEEE Geoscience and Remote Sensing Letters,* vol. 16, 7, pp. 1031-1035, 2019.
48. H. Zhu *et al.*, "Automatic multilabel electrocardiogram diagnosis of heart rhythm or conduction abnormalities with deep learning: a cohort study," *The Lancet. Digital health,* vol. 2, no. 9, pp. e348-e357, 2020.

49. N. Navamajiti, T. Saethang, and D. Wichadakul, "McBel-Plnc: A Deep Learning Model for Multiclass Multilabel Classification of Protein-lncRNA Interactions," presented at the Proceedings of the 2019 6th International Conference on Biomedical and Bioinformatics Engineering (ICBBE'19), Shanghai, China, 2019.
50. B. Namazi, G. Sankaranarayanan, and V. Devarajan, "LapTool-Net: A Contextual Detector of Surgical Tools in Laparoscopic Videos Based on Recurrent Convolutional Neural Networks," *ArXiv,* vol. abs/1905.08983, 2019.
51. X. Zhou, Y. Li, and W. Liang, "CNN-RNN Based Intelligent Recommendation for Online Medical Pre-Diagnosis Support," *IEEE/ACM Transactions on Computational Biology and Bioinformatics,* vol. 18, no. 3, pp. 912-921, 2021.
52. A. E. Samy, S. R. El-Beltagy, and E. Hassanien, "A Context Integrated Model for Multi-label Emotion Detection," *Procedia Computer Science,* vol. 142, pp. 61-71, 2018/01/01/ 2018, doi: https://doi.org/10.1016/j.procs.2018.10.461.
53. J. Zhang, Q. Chen, and B. Liu, "NCBRPred: predicting nucleic acid binding residues in proteins based on multilabel learning," *Briefings in bioinformatics,* vol. 22, no. 2, 2021.
54. D. Li, H. Wu, J. Zhao, Y. Tao, and J. Fu, "Automatic Classification System of Arrhythmias Using 12-Lead ECGs with a Deep Neural Network Based on an Attention Mechanism," *Symmetry,* vol. 12, no. 11, p. 1827, 2020. [Online]. Available: https://www.mdpi.com/2073-8994/12/11/1827.
55. J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, "Comprehensive comparative study of multi-label classification methods," *Expert Systems with Applications,* vol. 203, p. 117215, 2022/10/01/ 2022, doi: https://doi.org/10.1016/j.eswa.2022.117215.
56. D. Turnbull, L. Barrington, D. A. Torres, and G. Lanckriet, "Semantic Annotation and Retrieval of Music and Sound Effects," *IEEE Transactions on Audio, Speech, and Language Processing,* vol. 16, no. 22, pp. 467-476, 2008, doi: 10.1109/TASL.2007.913750.
57. M.-L. Zhang and Z. Zhou, "ML-KNN: A lazy learning approach to multi-label learning," *Pattern Recognit.,* vol. 40, no. 7, pp. 2038-2048, 2007, doi: 10.1016/j.patcog.2006.12.019.
58. L. Chen, "Predicting anatomical therapeutic chemical (ATC) classification of drugs by integrating chemical-chemical interactions and similarities," *PLoS ONE,* vol. 7, no. e35254, 2012.
59. K. C. Chou, "Some remarks on predicting multi-label attributes in molecular biosystems," *Molecular Biosystems,* vol. 9, pp. 10922-1100, 2013.
60. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.
61. F. Gers, J. Schmidhuber, and F. Cummins, "Learning to Forget: Continual Prediction with LSTM," *Neural Computation,* vol. 12, no. 10, pp. 2451-2471, 2000, doi: 10.1162/089976600300015015.
62. Y. Su, Y. Huang, and C.-C. J. Kuo, "On Extended Long Short-term Memory and Dependent Bidirectional Recurrent Neural Network," *Neurocomputing,* vol. 356, pp. 151–161, 2019.
63. J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling," *ArXiv,* vol. abs/1412.3555, 2014.
64. L. Jing *et al.*, "Gated Orthogonal Recurrent Units: On Learning to Forget," *Neural Computation,* vol. 31, no. 4, pp. 765-783, 2019, doi: doi.org/10.1162/neco_a_01174.
65. K. Zhang, Z. Liu, and L. Zheng, "Short-Term Prediction of Passenger Demand in Multi-Zone Level: Temporal Convolutional Neural Network With Multi-Task Learning," *IEEE Transactions on Intelligent Transportation Systems,* vol. 21, no. 4, pp. 1480-1490, 2020, doi: 10.1109/TITS.2019.2909571.
66. A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comp Surv,* vol. 31, no. 3, pp. 264-323, 1999.
67. S. Dubey, S. Chakraborty, S. K. Roy, S. Mukherjee, S. K. Singh, and B. Chaudhuri, "diffGrad: An Optimization Method for Convolutional Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 31, no. 11, pp. 4500-4511, 2020.
68. L. Nanni, G. Maguolo, and A. Lumini, "Exploiting Adam-like Optimization Algorithms to Improve the Performance of Convolutional Neural Networks," *ArXiv,* vol. abs/2103.14689, 2021.
69. L. Nanni, A. Manfe, G. Maguolo, A. Lumini, and S. Brahnam, "High performing ensemble of convolutional neural networks for insect pest image detection," *ArXiv,* vol. abs/2108.12539, 2021.
70. L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," *2017 IEEE Winter Conference on Applications of Computer Vision (WACV),* pp. 464-472, 2017.
71. X. Cheng, W.-Z. Lin, X. Xiao, and K.-C. Chou, "pLoc_bal-mAnimal: predict subcellular localization of animal proteins by balancing training dataset and PseAAC," *Bioinformatics,* vol. 35, no. 3, pp. 398-406, 2018, doi: 10.1093/bioinformatics/bty628.
72. M. Liu *et al.*, "Large-scale prediction of adverse drug reactions using chemical, biological, and phenotypic properties of drugs," *Journal of the American Medical Informatics Association : JAMIA,* vol. 19, pp. e28 - e35, 2012, doi: doi:10.1136/amiajnl-2011-000699.
73. L. Yang, X.-Z. Wu, Y. Jiang, and Z. Zhou, "Multi-Label Learning with Deep Forest," *ArXiv,* vol. abs/1911.06557, 2020.
74. F. K. Nakano, K. Pliakos, and C. Vens, "Deep tree-ensembles for multi-output prediction," *Pattern Recognit,* vol. 121, p. 108211, 2022/01/01/ 2022, doi: https://doi.org/10.1016/j.patcog.2021.108211.
75. X. Fu, D. Li, and Y. Zhai, "Multi-label learning with kernel local label information," *Expert Systems with Applications,* vol. 207, p. 118027, 2022/11/30/ 2022, doi: https://doi.org/10.1016/j.eswa.2022.118027.
76. Z. B. Yu and M. L. Zhang, "Multi-Label Classification With Label-Specific Feature Generation: A Wrapped Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 44, no. 9, pp. 5199-5210, 2022, doi: 10.1109/TPAMI.2021.3070215.