**Multi Label Risk Prediction Diabetes Complication using Deep Neural Network with Multi-Channel Weighted Dropout**

1,2Nur Rachman Dzakiyullah, 1M. A. Burhanuddin, 1Raja Rina Raja Ikram, 3Novanto Yudistira, 3Muhammad Rifqi Fauzi, Mustafa Musa

1Faculty of Information & Communication Technology, Universiti Teknikal Malaysia Melaka, Durian Tunggal, Melaka

2Faculty of Computer and Engineering, Department of Information System, Universitas Alma Ata, Yogyakarta, Indonesia

3Faculty of Computer Science, University of Brawijaya, Indonesia

**Abstract**

This paper briefly introduces an enhanced neural network regularization method, so called weight dropout, in order to prevent deep neural networks from overfitting. In suggested method, the fully connected layer jointly used with weight dropout is a collection of layers in which the weights between nodes are dropped randomly on the process of training. To accomplish the desired regularization method, we propose a building blocks with our weight dropout mask and DNN. The performance of proposed method has been compared with other previous methods in the domain of healthcare which is risk prediction diabetes complication for the evaluation purpose. The results show that the proposed method gives successful performance.

Keyword:Deep Neural Network, Multi-Channel Weighted Dropout, Risk Prediction Diabetes Complication, Multi-Label Classification

# 1. INTRODUCTION

Deep Learning (DL) is part of machine learning techniques have gained prominence in various classification task and practical implications. The advance technology and computing power today’s give advantage to lead modern DL while achieving great performance with higher accuracy in order to solving task in domains context or real-world application.

Dropout is a regularization method used to prevent overfitting in neural networks. Overfitting occurs when a model learns to perform well on the training data but fails to generalize to new, unseen data. Dropout helps combat overfitting by randomly dropping out (i.e., setting to zero) a fraction of the neurons during each forward and backward pass of training.

[1] Overfitting is a common problem in machine learning, which means the model too closely fits the training data while performing poorly in the test data. Among various methods of coping with overfitting, dropout is one of the representative ways. From randomly dropping neurons to dropping neural structures, dropout has achieved great success in improving model performances. Although various dropout methods have been designed and widely applied in past years, their effectiveness, application scenarios, and contributions have not been comprehensively summarized and empirically compared by far. It is the right time to make a comprehensive survey. OVERFITTING is a common problem in the training pro- cess of neural network models [1]. Due to the large number of parameters and strong fitting ability, most neural models perform well on the training set, while they may perform poorly on the test set. Some methods have been proposed in previous studies to address the overfitting problem, such as adding a regularization term to penalize the total size of model parameters [2] and applying Batch Normalization [3] or Weight Normalization [4] to regularize deep neural networks.

[2]Dropout [7,17] is arguably the most popular strategy [3,4] to avoid overfitting in neural networks. However, there are several shortcomings of this technique. First, connections to neurons are dropped out randomly. As a result, some neurons that may have produced or propagated class discriminative attributes may be dropped out leading to loss of information [23] . Second, we have no means to control dropout based on the performance of the network during training. The role of a neuron-pair, connected by a weight, is not exploited to drop out the connection between neuron-pair. Therefore, dropout is not dependent on the complex interplay of neurons between a pair of network layers.

However, DNNs have also been shown to be making mostly over-confidentpredictions [15], a side-effect of the heuristics used in modern DNNs. This means that for ambiguous instances bordering two classes (e.g., human wearing a cat costume), or on unrelated instances (e.g., plastic bag not “seen” during training and classified with high probability as rock), DNNs are likely to fail silently, which is a critical drawback for decision making systems. This has motivated several works to address the predictive uncertainty of DNNs [6, 10, 31], usually taking inspiration from Bayesian approaches. Knowledge about the distribution of the network weights during training opens the way for studying the evolution of the underlying covariance matrix, and the uncertainty of the model parameters, referred to as the epistemic uncertainty [26]. In this work we propose a method for estimating the distribution of the weights by tracking their trajectory during training. This enables us to sample an ensemble of networks and estimate more reliably the epistemic uncertainty and detect out-of-distribution samples.

To improve DNN model using **Multi-Channel Weighted Dropout**

Dropout is a regularization technique used in neural networks to prevent overfitting and improve generalization performance. It involves randomly dropping (setting to zero) a proportion of neurons during training. Here are several dropout techniques commonly used in neural networks:

Standard Dropout:

During each training iteration, a random subset of neurons is dropped out (i.e., their outputs are set to zero) with a specified probability.

This helps prevent co-adaptation of neurons and encourages the network to learn more robust features.

Spatial Dropout:

Applied specifically to convolutional layers in convolutional neural networks (CNNs).

Instead of dropping out individual neurons, entire feature maps are dropped out together.

This helps maintain the spatial relationships between the features.

Dropout on Input Layers:

In addition to applying dropout on hidden layers, dropout can also be applied to the input layer.

This helps prevent overfitting by preventing the network from relying too heavily on specific input features.

DropConnect:

Similar to dropout but applied to connections (weights) rather than neurons.

Randomly sets a proportion of weights to zero during each training iteration.

This can be computationally expensive, but it provides a form of regularization.

Zoneout:

Used in recurrent neural networks (RNNs).

Randomly sets entire hidden states of neurons to zero during training, similar to dropout in feedforward networks.

It helps prevent overfitting in the context of sequential data.

DropBlock:

A technique inspired by spatial dropout and applied to convolutional layers.

Instead of dropping individual pixels, entire contiguous blocks of pixels are dropped from feature maps.

This is particularly useful in preventing the learning of fine-grained details that may be noisy.

Scheduled Dropout:

The dropout rate is gradually increased during training.

This starts with a low dropout rate and increases it over time, allowing the model to initially focus on learning and later introduce more regularization.

Alpha Dropout:

A variant of dropout specifically designed for the rectified linear unit (ReLU) activation function.

It maintains the statistics of the activation function during training to ensure proper scaling.

These dropout techniques are aimed at preventing overfitting, improving generalization, and enhancing the robustness of neural networks. Depending on the specific architecture and task, different dropout techniques may be more suitable.

[3] Improving neural networks by preventing co-adaptation of feature detectors

[4]On Dropout, Overfitting, and Interaction Effects in Deep Neural Networks

[5]

[6] Dropout: A simple way to prevent neural networks from overfitting

[7] LocalDrop: A Hybrid Regularization for Deep Neural Networks

[8] Improving deep neural networks by using sparse dropout strategy

[9] Automatically classifying non-functional requirements using deep neural network

[10] Continuous Dropout

Dropout as optimization [11] Understanding dropout as an optimization trick

[12] DropELM: Fast neural network regularization with Dropout and DropConnect

[13] EDropout: Energy-Based Dropout and Pruning of Deep Neural Networks

[14] Controlled dropout: A different dropout for improving training speed on deep neural network

[15]Acceleration of DNN training regularization: Dropout accelerator

[16] Improved Convolutional Neural Network Fault Diagnosis Method Based on Dropout

[17] Concrete Dropout

[18] Fast dropout training

[19] Dropout as a Bayesian approximation: Representing model uncertainty in deep learning

[20] Rademacher dropout: An adaptive dropout for deep neural network via optimizing generalization gap

[21] The dropout learning algorithm

[22] Quality of randomness and node dropout regularization for fitting neural networks

[23] Adaptive sparse dropout: Learning the certainty and uncertainty in deep neural networks

[24] Dropout Reduces Underfitting

[25] Uncertainty propagation for dropout-based Bayesian neural networks

[26] An Overview of Overfitting and its Solutions

There is several way in using Dropout in order to improve DNN.

Multi sample dropout,

Komparasi dengan tanpa dropout, komparasi dropout biasa,

Aplikasi dropout weighted ke AA-MLC di komparasi dengan hasil DNN biasa yang hyper.

Apa yang bisa menunjukkan keunggulan metode dropout weighted ini?

# 2. RELATED WORKS

The multi-sample dropout regularization technique presented in this paper can achieve better generalization and faster training than the original dropout. Dropout is one of the most widely used regularization techniques, but a wide variety of other regularization techniques for better generalization have been reported. They include, for example, weight decay [15], data augmentation [3], [2], [24], [11], label smoothing [21], and batch normalization [12]. Although batch normalization is aimed at accelerating training, it also improves generalization. Many of these techniques are network independent while others, such as Shake-Shake [5] and Drop-Path [16], are specialized for a specific network architecture. The success of dropout led to the development of many variations that extend the basic idea of dropout (e.g. [6], [10], [22], [4]. The techniques reported use a variety of ways to randomly drop information in the network. For example, DropConnect [23] discards randomly selected connections between neurons. DropBlock [6] randomly discards areas in convolution layers while dropout is typically used in fully connected layers after the convolution layers. Stochastic Depth [10] randomly skip layers in a very deep network. However, none of these techniques use the approach used in our multi-sample dropout. Many of them can be used with multi sampling technique to make the divergence among dropout samples. Another way to enhance the dropout is adaptively tuning the dropout ratio (e.g. [1]. These techniques are also orthogonal to the multi sampling technique, since the multi-sample dropout does not depend on specific dropout ratio as we have already shown. Multi-sample dropout calculates the final prediction and loss by averaging the results from multiple loss functions. Several network architectures have multiple exits with loss functions. For example, GoogLeNet [20] has two early exits in addition to the main exit, and the final prediction is made using a weighted average of the outputs from these three loss functions. Unlike multi-sample dropout, GoogLeNet creates the two additional exits at earlier positions in the network. Multi-sample dropout creates multiple uniform exits, each with a loss function, by duplicating a part of the network.

[27]In the years since, a wide range of stochastic techniques inspired by the original dropout method have been proposed for use with deep learning models. We use the term dropout methods to refer to them in general. They include dropconnect [3], standout [4], fast dropout [5], variational dropout [6], Monte Carlo dropout [7] and many others. Generally speaking, dropout methods involve randomly modifying neural network parameters or activations during training or inference, or approximating this process. Figure 1 illustrates research into dropout methods over time.

While originally used to avoid overfitting, dropout methods have since expanded to a variety of ap- plications. The two additional applications discussed in this paper are the use of dropout to compress deep neural networks [8–11] and Monte Carlo dropout [7], which measures the uncertainty of deep learning models during inference.

Survey Dropout [1] A Survey on Dropout Methods and Experimental Verification in Recommendation, [27] Survey of Dropout Methods for Deep Neural Networks

[28] Weight Dropout for Preventing Neural Networks from Overfitting

[29] Advanced Dropout: A Model-Free Methodology for Bayesian Dropout Optimization

[30] A novel trilinear deep residual network with self-adaptive Dropout method for short-term load forecasting

[31] Jumpout: Improved dropout for deep neural networks with RELus

[32]Ising-Dropout: A Regularization Method for Training and Compression of Deep Neural Networks

[33]Application of three statistical models for predicting the risk of diabetes

Risk Prediction of Disease Complications in Type 2 Diabetes Patients Using Soft Computing Techniques

[34]An ensemble approach for classification and prediction of diabetes mellitus using soft voting classifier

[35]Predicting the onset of type 2 diabetes using wide and deep learning with electronic health records

[36] One-vs-One classification for deep neural networks

[37]Type 2 diabetes mellitus prediction model based on data mining

[38] Building Risk Prediction Models for Type 2 Diabetes Using Machine Learning Techniques

[39]

[40] RR-interval signals known as heart rate variability (HRV) signals (derived from electrocardiogram (ECG) signals) can be effectively used for the non-invasive detection of diabetes

[41] Diabetes prediction model based on an enhanced deep neural network

In this study we proposed, methodology for

[42] The Methodology for Diabetes Complications Prediction Model

Adaptive Dropout

[43]

# 3. PROPOSED METHOD

In this study Proposed Multi-Channel Weighted Dropout for improving DNN model performance while designing risk prediction diabetes complication.

This section describes the multi-sample dropout technique. The basic idea is quite simple: create multiple dropout samples instead of only one. Figure 1 depicts an easy way to implement multi-sample dropout (with two dropout samples) using an existing deep learning framework with only common opera-tors. The dropout layer and several layers after the dropout are duplicated for each dropout sample; in the figure, the ”dropout,” ”fully connected,” and ”softmax + loss func” layers are duplicated. Different masks are used for each dropout sample in the dropout layer so that a different subset of neurons is used for each dropout sample. In contrast, the parameters (i.e., connection weights) are shared between the duplicated fully connected layers. The loss is computed for each dropout sample using the same loss function, e.g., cross entropy, and the final loss value is obtained by averaging the loss values for all dropout samples. This final loss value is used as the objective function for optimization during training. We select the class label as the prediction based on the average of outputs from the last fully connected layer. Although a configuration with two dropout samples is shown in Figure 1, multi-sample dropout can be configured to use any number of dropout samples. The original dropout can be seen as a special case of multi-sample dropout where the number of samples is set to one.

During inference, neurons are not discarded as is done in the original dropout. The loss can be calculated using only one dropout sample because the dropout samples become identical at the inference time if we do not drop any neurons at the dropout layer. Hence, we always use only one dropout sample at inference regardless of the training method. Compared to Importance Weighted Stochastic Gradient Descent (IWSGD) [18], which also makes multiple samples by dropout, we only duplicate operations in a part of the forward pass after the dropout while IWSGD duplicate operations in the entire backward pass as well as the forward pass. Hence our multi-sample dropout is much more light weight in terms of computation costs. Especially when dropout is applied to a layer near the end of the network, the additional execution time due to the duplicated operations in multi- sample dropout is not significant; this characteristic makes multi-sample dropout more suitable for deep CNNs. We can apply multi-sample dropout for shallow networks, such as the multilayer perceptron. We observed that multi-sample dropout reduces the number of iterations for training even for shallow networks, but the costs of the increased execution time per iteration surpassed the benefits; due to the increase in the computation time per iteration, multi-sample dropout actually degraded the training speed in terms of the computation time. If the network includes multiple dropout layers, we can apply multi-sample dropout at any of these dropout layers. Multi sampling at an earlier dropout layer may increase diversity among dropout samples and increase the benefits in trade for the higher additional costs due to more duplicated layers.

Why multi-sample dropout accelerates training

Intuitively, the effect of multi-sample dropout with M dropout samples is similar to that of enlarging the size of a minibatch M times by duplicating each sample in the mini- batch M times, e.g. Batch Augmentation [9]. For example, if a minibatch consists of two data samples ?A,B?, training a network by using multi-sample dropout with two dropout samples closely corresponds to training a network by using the original dropout and a minibatch of ?A, A,B, B? assuming a different mask applied to each sample in the minibatch. This is similar to batch augmentation [9], which applies a different data augmentation for each of duplicated samples to make diversity among duplicated samples. Using a larger minibatch size with duplicated samples may not make sense to accelerate the training because it increases the computation time per iteration by M times. In contrast, multi-sample dropout can enjoy similar gains without a huge increase in computation cost per iteration for deep CNNs because it duplicates only the operations after dropout. For example, when we duplicate the last two fully connected layers of VGG16 [19] eight times, we observed the increased execution time per iteration by only 2%. Because of the non-linearity of the activation functions, the original dropout with duplicated samples and multi-sample dropout do not give exactly the same results. However, similar acceleration was observed in the training in terms of the number of iterations, as shown by the experimental results.

C. Why multi-sample dropout yields higher accuracy

Noh et al. [18] showed that creating multiple samples during the training of deep networks improves the accuracy of the trained network. Training of a noisy network (e.g. with dropout) requires optimizing the marginal likelihood over the noise ( and SGD optimizers optimize the network using approximated marginal likelihood based on the finite number of samples (LSGD) as its objective function. Here, the SGD objective function LSGD is the lower-bound of the marginal likelihood over the noise and using more dropout samples makes the lower-bound tighter, i.e. here, LSGD(M) means LSGD when M dropout samples are used. This results in better accuracy in the trained network with increasing number of dropout samples. Although Noh’s Importance Weighted Stochastic Gradient Descent (IWSGD) makes multiple noise (dropout) samples at dropout like our multi-sample dropout, it executes both the forward pass and backward pass separately for each sample, and then calculates the gradients for updating network parameters as weighted av- erage of the gradients calculated for each dropout sample with the normalized likelihood for the sample as the weight. Our results showed that much simpler and light-weight technique which only duplicates a small part of the forward pass can enjoy the benefits of using multiple dropout samples.

D. Other sources of diversity among samples

The key to faster training with multi-sample dropout is the diversity among dropout samples; if there is no diversity, the multi-sampling technique gives no gain and simply wastes computation resources. Although we tested only dropout in this paper, the multi-sampling technique can be used with other sources of diversity. For example, variants of dropout, such as DropConnect, can be enhanced by using the multi-sampling technique.

|  |
| --- |
| Proposed MCWD  Standard Dropout |
| Figure 3.1 Overview of Proposed Multi-Channel Weighted Dropout Vs Standard dropout |

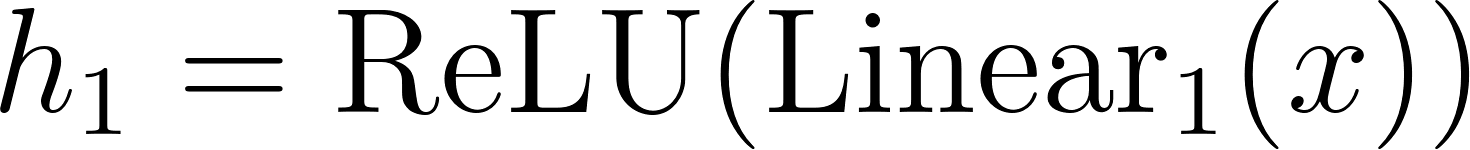
How weighted dropout works:

1. The input of the 3-channel dropout will be divided into 3 so that it will become 1 channel
2. Each channel is put into the dropout layer with a probability of 0.3 e.g.
3. The result of the dropout of each channel will be multiplied by the weight of the previous layer (Fully-Connected layer) by adjusting the weight of the respective channel. For example, the first channel dropout result is multiplied by the weight of the first channel as well.
4. Then the weighted dropout results from several channels will be combined into the original form as before entering the dropout
5. The merged result goes into the next layer

|  |
| --- |
| **Proposed Deep Neural Network with Multi-Channel Weighted Dropout** |
| **# Define a neural network model with multi-channel weighted dropout**  class MCWD:  **# Constructor to initialize the model**  function initialize(num\_features=26, num\_classes=7, dropout\_probabilities=[0.05, 0.01,   0.003]):  **# First layer with 32 neurons and ReLU activation**  first\_layer = create\_dense\_layer(32, activation='relu', input\_shape=(num\_features,))  **# Second layer with softmax activation, the length is based on dropout probabilities**  second\_layer = create\_dense\_layer(length(dropout\_probabilities), activation='softmax')  **# Initialize the dropout layers based on provided probabilities**  dropout\_layers = [create\_dropout\_layer(prob) for prob in dropout\_probabilities]  **# Third layer with sigmoid activation for the final output**  third\_layer = create\_dense\_layer(num\_classes, activation='sigmoid')  **# Method to perform forward pass through the model**  function forward\_pass(input\_data, training\_mode=None, mask=None):  **# Pass input data through the first layer**  output\_first\_layer = process\_through\_layer(input\_data, first\_layer)  **# Save the weights from the first layer**  weights = output\_first\_layer  **# Apply weighted dropout to each channel**  for i, dropout\_layer in enumerate(dropout\_layers):  # Apply dropout to the channel and multiply with weights  modified\_channel = apply\_weighted\_dropout(output\_first\_layer[:, i], dropout\_layer,   weights[:, i], training\_mode)  **# Stack the modified channels back together**  modified\_channels\_stacked = stack\_channels(modified\_channel)  **# Pass the modified channels through the third layer**  final\_output = process\_through\_layer(modified\_channels\_stacked, third\_layer)  return final\_output |

Let's represent the operations in the provided PyTorch code as mathematical equations:

1. First layer (self.fcn1):

- [](https://www.codecogs.com/eqnedit.php?latex=h_1%20%3D%20%5Ctext%7BReLU%7D(%5Ctext%7BLinear%7D_1(x))#0)

- [](https://www.codecogs.com/eqnedit.php?latex=h_2%20%3D%20%5Ctext%7BReLU%7D(%5Ctext%7BLinear%7D_2(h_1))#0)

- [](https://www.codecogs.com/eqnedit.php?latex=h_3%20%3D%20%5Ctext%7BSoftmax%7D(%5Ctext%7BLinear%7D_3(h_2))#0)

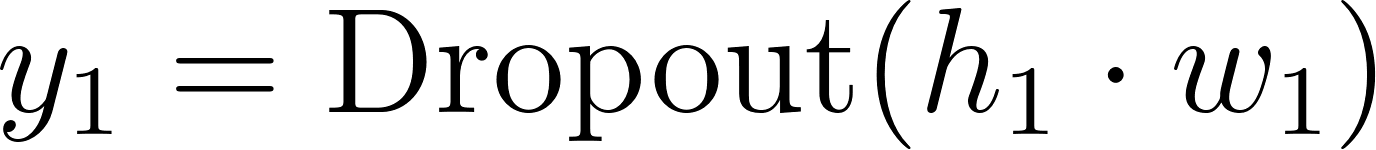
- [A black background with a black square

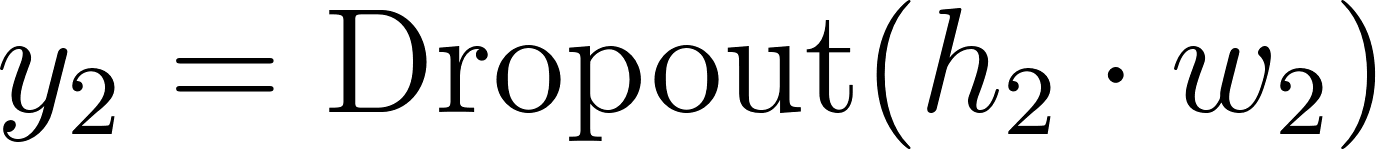
Description automatically generated with medium confidence](https://www.codecogs.com/eqnedit.php?latex=x_%7B%5Ctext%7Bout%7D%7D%20%3D%20%5Bh_1%2C%20h_2%2C%20h_3%5D#0)

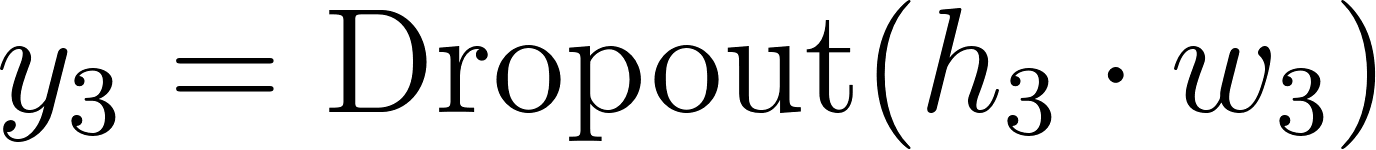
2. Weighted Dropout:

- [A black background with a black square

Description automatically generated with medium confidence](https://www.codecogs.com/eqnedit.php?latex=w%20%3D%20x_%7B%5Ctext%7Bout%7D%7D#0) (save the weights)

- [](https://www.codecogs.com/eqnedit.php?latex=y_1%20%3D%20%5Ctext%7BDropout%7D(h_1%20%5Ccdot%20w_1)#0)

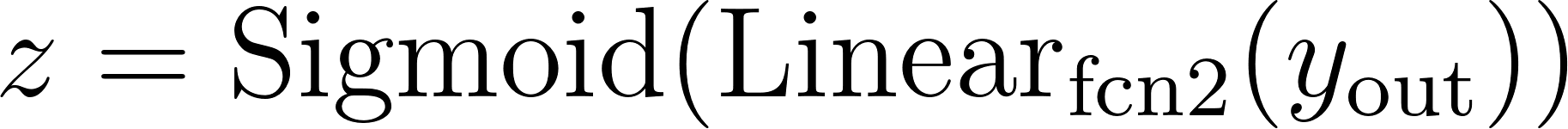
- [](https://www.codecogs.com/eqnedit.php?latex=y_2%20%3D%20%5Ctext%7BDropout%7D(h_2%20%5Ccdot%20w_2)#0)

- [](https://www.codecogs.com/eqnedit.php?latex=y_3%20%3D%20%5Ctext%7BDropout%7D(h_3%20%5Ccdot%20w_3)#0)

- [A black background with a black square

Description automatically generated with medium confidence](https://www.codecogs.com/eqnedit.php?latex=y_%7B%5Ctext%7Bout%7D%7D%20%3D%20%5By_1%2C%20y_2%2C%20y_3%5D#0)

3. Second layer (self.fcn2):

- [](https://www.codecogs.com/eqnedit.php?latex=z%20%3D%20%5Ctext%7BSigmoid%7D(%5Ctext%7BLinear%7D_%7B%5Ctext%7Bfcn2%7D%7D(y_%7B%5Ctext%7Bout%7D%7D))#0)

In summary, the mathematical representation of the given PyTorch code is:

[A close-up of a letter

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=h_1%20%26%3D%20%5Ctext%7BReLU%7D(%5Ctext%7BLinear%7D_1(x))#0)

[A close-up of a letter

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=h_2%20%26%3D%20%5Ctext%7BReLU%7D(%5Ctext%7BLinear%7D_2(h_1))#0)

[A black text with a black background

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=h_3%20%26%3D%20%5Ctext%7BSoftmax%7D(%5Ctext%7BLinear%7D_3(h_2))#0)

[A black and white symbol

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=x_%7B%5Ctext%7Bout%7D%7D%20%26%3D%20%5Bh_1%2C%20h_2%2C%20h_3%5D#0)

[A black and white image of a sign

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=w%20%26%3D%20x_%7B%5Ctext%7Bout%7D%7D#0)

[](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=y_1%20%26%3D%20%5Ctext%7BDropout%7D(h_1%20%5Ccdot%20w_1)#0)

[A black and white logo

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=y_2%20%26%3D%20%5Ctext%7BDropout%7D(h_2%20%5Ccdot%20w_2)#0)

[A black and white logo

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=y_3%20%26%3D%20%5Ctext%7BDropout%7D(h_3%20%5Ccdot%20w_3)#0)

[A black and white letter y

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=y_%7B%5Ctext%7Bout%7D%7D%20%26%3D%20%5By_1%2C%20y_2%2C%20y_3%5D#0)

[A black and white text

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=z%20%26%3D%20%5Ctext%7BSigmoid%7D(%5Ctext%7BLinear%7D_%7B%5Ctext%7Bfcn2%7D%7D(y_%7B%5Ctext%7Bout%7D%7D))#0)

Note: The notation [A black and white text

Description automatically generated](http://www.sciweavers.org/tex2img.php?bc=Transparent&fc=Black&im=jpg&fs=100&ff=modern&edit=0&eq=h_i%24%20represents%20the%20output%20of%20the%20#0)i[A black background with a black square

Description automatically generated with medium confidence](https://www.codecogs.com/eqnedit.php?latex=-th%20linear%20layer%2C%20#0)w\_i[](https://www.codecogs.com/eqnedit.php?latex=%20represents%20the%20weights%20obtained%20from%20the%20corresponding%20output%2C%20and%20#0)y\_i[](https://www.codecogs.com/eqnedit.php?latex=%20represents%20the%20output%20after%20applying%20weighted%20dropout%20on%20#0)h\_i$$. The final output is obtained after the second linear layer and sigmoid activation.

# 4. EXPERIMENTAL DESIGN AND RESULT

In this section, we discuss the design methodology and the result of proposed method experimentation details. In Figure 4.1 below show general methodology has several step that going through the experiments details. **First phase**, we present the data set used that conduct in this study.

|  |
| --- |
|  |
| Figure 4.1 Design Methodology |

The dataset utilized is The Behavioral Risk Factor Surveillance System (BRFSS) from 2016-2021. It is a prominent system of health-related telephone surveys that gather state data on U.S. residents' health-related risk behaviors, chronic health conditions, and utilization of preventive services [44]. However, this study only examines 33 variables from each year of BRFSS data. This study is an extension of the research conducted in reference [38], which examined several factors associated with type 2 diabetes. Additionally, it investigates factors linked to diabetes complications as outlined in reference [45] to develop risk prediction model diabetes complications based on our proposed method. **Appendix 1** has a list of 33 things that were investigated as potential risk factors for diabetic complications in this study. The BRFSS dataset contains variables related to diabetes complications including Nephropathy, Coronary Heart Disease, Heart Attack Disease, Stroke, Cancer, Arthritis, and Depression. The dataset includes individuals with diabetes, prediabetes, and no dependent variable. When designing the Risk Prediction Diabetes Complication model, factors including physical and mental health, healthcare coverage and provider, and level of urbanization will be considered as additional risk indicators. The independent variables consist of Nephropathy, Coronary Heart Disease, Heart Attack Disease, Stroke, Cancer, Arthritis, and Depression as multi-label. Our goal was to determine the most effective way for predicting diabetic complications to aid in personalized type 2 diabetes treatment and inform therapeutic choices using our proposed approach.

**The second and third phases** involve data analysis along with data pre-processing, a crucial step for obtaining a more profound comprehension of the data. The BFFRS dataset has a total of 2,632,674 rows and 33 columns. Each variable has been renamed as X\_input (X1-X26) and Y\_output (Y1-Y2). Data cleaning was performed to detect and remove erroneous data or noise, and to handle missing information by removing rows or columns with NaN values. Utilize mean imputation methods to handle missing data in this study. The scikit-learn library contains a class named "Iterative Imputer" that use the MICE (Multiple Imputation by Chained Equations) method for imputation. This method creates a regression model for each missing feature. The model's estimated values are used to fill in missing feature values until convergence or the maximum number of iterations is reached. Every missing attribute is handled by creating a regression model using this approach. The model's predicted values are used to fill in missing feature values until convergence or the maximum number of iterations is reached.The "Iterative Imputer" is configured with defined parameters such as missing\_values (NaN), max\_iter (maximum iterations), tol (convergence tolerance), n\_nearest\_features (number of nearest features for imputation), and initial\_strategy (strategy for initializing missing values). The imputer is used on the Data Frame to replace missing values using the transform method, and the result is stored in the imputed\_data DataFrame. The BRFSS dataset has various issues that require resolution, such as useless data values such responses of "don't know" or refusals to answer. The BRFSS dataset categorized age groups as follows: 1 for 31 to 40 years, 2 for 41–50 years, 3 for 51–60 years, 4 for 61–70 years, 5 for 71–80 years, and 6 for over 81 years, in relation to mental health.

**The fourth phase** is the step on experimental design, which discusses details. The study utilized several settings on MLC Frameworks such as MLC-Problem Transformation (MLC-PT) and MLC-Algorithm Adaptation (MLC-AA) to develop risk prediction models for diabetes complications. According to [46]–[48] Multi-label categorization requires transformation into distinct learning problems. First-order, second-order, and high-order approaches are exemplary. The technique is determined by the multi-label learning problem and the desired balance between computing efficiency and correlation modelling. The first-order technique, while straightforward and effective, may overlook label connections. The second-order technique involves paired relations and has good generalization capabilities, but it may overlook higher-order correlations. The high-order technique is comprehensive but entails increased computational complexity and may involve ensemble methods, meta-learning, or other intricate procedures that beyond basic modifications. Various techniques are suitable based on the multi-label learning issue. The study employed Binary Relevance (BR), Calibrated Label Ranking (CLR), Classifier Chains (CC), and Label Powerset (LP) as techniques for MLC-PT. Moreover, MLC-AA directly addresses multi-label learning problem, unlike MLC-PT. Algorithm adaptation refers to the process of modifying algorithms to suit certain requirements. To understand on how MLC frameworks are conducted in this study we explain more detail regarding the fourth phase into several scenario.

In this phase explain three experimental scenario to achieve our result. In experimental **scenario one**, this study do comparison in the context of MLC Framework to identify whichsuitable framework for our proposed methods. To compare each MLC Framework, we are comparing DNN model using standard dropout vs Proposed method. The probability dropout is set equal 0.3 and compare time training and testing for each model. Commonly, the ratio probability dropout used between 0.1-0.3 [1]. The study utilized a value of 0.3 due to considerations related to model complexity, data properties, and overfitting. The dropout rate is a hyperparameter that specifies the likelihood of a neuron being dropped out in each layer. A higher dropout rate increases the probability of neurons being excluded, resulting in enhanced regularization. Neural networks with numerous parameters or layers are more prone to overfitting because of their ability to memorize the training data. Increasing the dropout rate enhances the network's capacity to acquire varied and autonomous characteristics, hence enhancing its capability to generalize to new input. This is crucial in scenarios with restricted data, high levels of noise, or intricate patterns. Neurons in a neural network tend to co-adapt during training, which can lead to them becoming too reliant on each other. Dropout disrupts these connections by randomly removing neurons, compelling the network to acquire more resilient and widespread representations. A larger dropout rate can serve as a more robust regularization technique, which helps prevent the network from being overly dependent on certain neurons or characteristics. This contributes to the development of a more resilient and universally applicable paradigm. Increasing the dropout rate is a common strategy to improve the generalization of complex models with many parameters and reduce overfitting.

**In scenario two**, similar with scenario one for standard dropout, however, our proposed model was set into three channel probability dropout along with weighted dropout with Different setting Probability Dropout 0.1, 0.2 and 0.3. The last but not least, **scenario three** was set the experiment on hyperparameter tunning the Proposed Model vs Standard Dropout to find best probability dropout. In this scenario three, all experiment used Hyperparameters Setting such as Loss Function: Binary Cross entropy; Optimizer: RMSProp; Learning Rate: 0.003; Epoch: 5; Batch Size: 32. This study compared all MLC frameworks using 20 performance evaluation parameters (**Appendix 2**) also AUC and ROC curves to evaluate model performance. According to [46]–[48], there is no general agreement about suitable evaluation matric for multi-label classification. The evaluation metrics for multi label learning can be divided into example-based metrics and label- based metrics. In our experimental settings, we conduct several evaluation metric that has been recommended in the previous study [46]–[48]. True labels are denoted by , while predicted labels for the same sample are denoted by . The ∆ symbol indicates a symmetric difference between sets, and represent examples and class labels, and , represent true positives and false positives for label, and and represent true negatives and false negatives for label . The accuracy and subset accuracy used in multi-label classification are example-based measures computed on the label set, unlike the general accuracy used in binary or multi-class classification. While macro average metrics calculate the metric for each class before averaging them, micro average metrics add up all class contributions to calculate the average metric. According to previous research [40], we evaluate the model using macro and micro criteria. Furthermore, the dataset was randomly split into the training (80%) and test (20%) sets. All studies then employed 10-fold cross validation to validate model performance and compare the time training along with testing time for computational Efficiency.

## 4.1 Experimental Result Analysis

In the MLC Frameworks setting, we labeled the experiments Nephropathy, Coronary Heart Disease, Heart Attack, Stroke, Cancer, Arthritis, and Depression. The objective in this study is to build the risk prediction diabetes complication by comparing DNN Model with standard Dropout and our proposed method that evaluated using 20 performance evaluation matric for MLC. The experimental result shown in Table 4.1 and Table 4.2 serve as empirical evidence for the study, supporting the discussion divided into three parts: (1) Comparison Proposed Model Vs Standard Dropout on Computational Efficiency with set equal Probability Dropout 0.3; (2) Comparison Proposed Model Vs Standard Dropout on Computational Efficiency with Different setting Probability Dropout; (3) Comparison Result on Example and Label based Evaluation of Hyperparameter Tunning for the Proposed Model vs. Standard Dropout, as follows:

Table 4.1 Result of Example Based Matric

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MLC Frameworks with DNN | | Example Based Matric | | | | | | | | | |
| **Sub-set Accuracy** | **Hamming Loss↓** | **Accuracy (exp)** | **Precision (exp)** | **Recall (exp)** | **F1-Score (exp)** | **One error↓** | **Coverage↓** | **Ranking Loss↓** | **Average precision** |
| PT-BR | Standard Dropout | 0,4141 | 0,1279 | 0,8621 | 0,9705 | 0,8838 | 0,9133 | 0,0720 | 6,6334 | 0,3543 | 0,8678 |
| Proposed (Same prob) | 0,4164 | 0,1271 | 0,8624 | 0,9661 | 0,8880 | 0,9124 | 0,0720 | 6,6159 | 0,3352 | 0,8693 |
| Proposed (Diff prob) | 0,4168 | 0,1274 | 0,8619 | 0,9643 | 0,8888 | 0,9117 | 0,0720 | 6,6159 | 0,3318 | 0,8693 |
| PT-CC | Standard Dropout | 0,4173 | 0,1272 | 0,8620 | 0,9642 | 0,8891 | 0,9119 | 0,0720 | 6,6187 | 0,3331 | 0,8697 |
| Proposed (Same prob) | 0,4169 | 0,1272 | 0,8616 | 0,9600 | 0,8925 | 0,9105 | 0,0720 | 6,5986 | 0,3160 | 0,8710 |
| Proposed (Diff prob) | 0,4173 | 0,1276 | 0,8616 | 0,9643 | 0,8886 | 0,9116 | 0,0720 | 6,6252 | 0,3343 | 0,8691 |
| PT-LP | Standard Dropout | 0,0043 | 0,4445 | 0,5231 | 0,5804 | 0,8431 | 0,5177 | 0,0720 | 6,9334 | 0,4405 | 0,8367 |
| Proposed (Same prob) | 0,0043 | 0,4445 | 0,5231 | 0,5804 | 0,8431 | 0,5177 | 0,0720 | 6,9334 | 0,4405 | 0,8367 |
| Proposed (Diff prob) | 0,0043 | 0,4445 | 0,5231 | 0,5804 | 0,8431 | 0,5177 | 0,0720 | 6,9334 | 0,4405 | 0,8367 |
| PT-CLR | Standard Dropout | 0,4146 | 0,1271 | 0,8615 | 0,9583 | 0,8937 | 0,9100 | 0,0720 | 6,5814 | 0,3092 | 0,8720 |
| Proposed (Same prob) | 0,4166 | 0,1270 | 0,8621 | 0,9624 | 0,8907 | 0,9114 | 0,0720 | 6,6027 | 0,3234 | 0,8706 |
| Proposed (Diff prob) | 0,4166 | 0,1270 | 0,8624 | 0,9653 | 0,8887 | 0,9123 | 0,0720 | 6,6059 | 0,3310 | 0,8700 |
| AA | Standard Dropout | 0,3938 | 0,1349 | 0,8365 | 0,9996 | 0,9868 | 0,9053 | 0,0700 | 6,2689 | 0,0894 | 0,9242 |
| Proposed (Same prob) | 0,4018 | 0,1326 | 0,8365 | 0,9996 | 0,9862 | 0,9079 | 0,0693 | 6,2655 | 0,0888 | 0,9270 |
| Proposed (Diff prob) | 0,4086 | 0,1296 | 0,8365 | 0,9996 | 0,9912 | 0,9118 | 0,0696 | 6,2544 | 0,0871 | 0,9293 |

**Notes:** and **↓** show the best performance

Table 4.2 Result of Label Based Matric

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MLC Frameworks with DNN | | Label Based Matric | | | | | | | | | |
| **Accuracy (micro)** | **Accuracy (macro)** | **Precision (micro)** | **Precision (macro)** | **Recall (micro)** | **Recall (macro)** | **F1-Score (micro)** | **F1-Score (macro)** | **AUC (micro)** | **AUC (macro)** |
| PT-BR | Standard Dropout | 0,8721 | 0,8721 | 0,8864 | 0,8732 | 0,9716 | 0,9601 | 0,9270 | 0,9133 | 0,6669 | 0,5530 |
| Proposed (Same prob) | 0,8729 | 0,8729 | 0,8904 | 0,8757 | 0,9670 | 0,9534 | 0,9272 | 0,9124 | 0,6787 | 0,5549 |
| Proposed (Diff prob) | 0,8726 | 0,8726 | 0,8914 | 0,8761 | 0,9654 | 0,9510 | 0,9269 | 0,9117 | 0,6813 | 0,5549 |
| PT-CC | Standard Dropout | 0,8728 | 0,8728 | 0,8916 | 0,8766 | 0,9654 | 0,9511 | 0,9270 | 0,9119 | 0,6819 | 0,5554 |
| Proposed (Same prob) | 0,8728 | 0,8728 | 0,8949 | 0,8793 | 0,9608 | 0,9445 | 0,9267 | 0,9105 | 0,6915 | 0,5568 |
| Proposed (Diff prob) | 0,8724 | 0,8724 | 0,8910 | 0,8757 | 0,9656 | 0,9511 | 0,9268 | 0,9116 | 0,6803 | 0,5545 |
| PT-LP | Standard Dropout | 0,5555 | 0,5555 | 0,8431 | 0,4818 | 0,5758 | 0,5714 | 0,6843 | 0,5177 | 0,5135 | 0,5000 |
| Proposed (Same prob) | 0,5555 | 0,5555 | 0,8431 | 0,4818 | 0,5758 | 0,5714 | 0,6843 | 0,5177 | 0,5135 | 0,5000 |
| Proposed (Diff prob) | 0,5555 | 0,5555 | 0,8431 | 0,4818 | 0,5758 | 0,5714 | 0,6843 | 0,5177 | 0,5135 | 0,5000 |
| PT-CLR | Standard Dropout | 0,8729 | 0,8729 | 0,8964 | 0,8813 | 0,9589 | 0,9420 | 0,9266 | 0,9100 | 0,6957 | 0,5579 |
| Proposed (Same prob) | 0,8730 | 0,8730 | 0,8933 | 0,8782 | 0,9633 | 0,9482 | 0,9270 | 0,9114 | 0,6868 | 0,5564 |
| Proposed (Diff prob) | 0,8730 | 0,8730 | 0,8913 | 0,8770 | 0,9661 | 0,9522 | 0,9272 | 0,9123 | 0,6811 | 0,5557 |
| AA | Standard Dropout | 0,8651 | 0,8651 | 0,8881 | 0,8713 | 0,9596 | 0,9426 | 0,9225 | 0,9053 | 0,8614 | 0,7722 |
| Proposed (Same prob) | 0,8674 | 0,8674 | 0,8877 | 0,8719 | 0,9633 | 0,9480 | 0,9240 | 0,9079 | 0,8635 | 0,7752 |
| Proposed (Diff prob) | 0,8704 | 0,8704 | 0,8863 | 0,8732 | 0,9695 | 0,9573 | 0,9261 | 0,9118 | 0,8666 | 0,7807 |

**Notes:** and **↓** showing the best performance

### Result on Scenario 1

From Table 4.1 and 4.2 show all the experimental result. To explain more detail regarding comparison Proposed Model Vs Standard Dropout on Computational Efficiency with set equal Probability Dropout 0.3 as scenario one that summarized in to Table 4.3 below:

Table 4.3 Comparison Proposed Model Vs Standard Dropout on Computational Efficiency with set equal Probability Dropout 0.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MLC-Framework with DNN** | | **Sum of Metrics** | **Time Training (Second) ↓** | **Time Testing (Second) ↓** |
| AA | Standard | 2 | **228.68** | 4.01 |
| Proposed (Same prob) | **18** | 231.10 | **3.94** |
| BR | Standard | 5 | **1,640.75** | **26.22** |
| Proposed (Same prob) | **15** | 1,678.57 | 27.76 |
| CC | Standard | 8 | **1,651.48** | **27.51** |
| Proposed (Same prob) | **12** | 1,706.18 | 28.13 |
| LP | Standard | 0 | **235.75** | **6.85** |
| Proposed (Same prob) | 0 | 253.36 | 7.63 |
| CLR | Standard | 8 | **3,078.49** | **151.07** |
| Proposed (Same prob) | **12** | 3,348.69 | 165.67 |

**Notes:** and **↓** showing the best performance

The result show that the proposed method was achieve promising result compare to standard dropout in the context of performance evaluation metric for all MLC Framework. MLC-AA with proposed method (same prob) has significant number of metric with 18 and followed by PT-BR with 15 metric; PT-CC and PT-CLR with 12 metric; and worse evaluation is PT-LP. However, in term of computation efficiency, our proposed model still not outperform compare with standard dropout. It’s because standard dropout architecture has more simpler architecture as shown in Figure 3.1 above. In addition, if we look at the difference in training time, our MLC-AA with proposed method is not significantly less, with a difference of 2.4 seconds between training time and the testing time 0.07 seconds.

### Result on Scenario 2

To explain more detail regarding result on Comparison Proposed Model Vs Standard Dropout on Computational Efficiency with Different setting Probability Dropout as scenario two can be seen in Table 4.4. From the result show that our proposed method outperformed even probability ratio was setting differently. The number of metric of MLC-AA is achieved significant compare with standard dropout. In this scenario two was analysed that our proposed model more robust even we change the setting into the channel. MLC-AA with proposed method (Diff prob) has significant number of metric achieved 18 and followed by PT-CC with Proposed (Same prob 0,3) achieved 12 metric; PT-CLR with proposed method (Diff prob) achieved 13 metric; PT-BR with proposed method (Diff prob) achieved 10 metric; and worse evaluation is PT-LP was same with previous scenario.

Table 4.4 Comparison Proposed Model Vs Standard Dropout on Computational Efficiency with Different setting Probability Dropout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MLC-Framework with DNN** | | **Sum of Metrics** | **Time Training (Second) ↓** | **Time Testing (Second) ↓** |
| AA | Standard (0,3) | 1 | **228.68** | 4.01 |
| Proposed (Same prob 0,3) | 1 | 231.10 | 3.94 |
| Proposed (Diff prob) | **18** | 231.52 | **3.82** |
| BR | Standard (0,3) | 5 | **1,640.75** | **26.22** |
| Proposed (Same prob 0,3) | 5 | 1,678.57 | 27.76 |
| Proposed (Diff prob) | **10** | 1,684.64 | 27.94 |
| CC | Standard (0,3) | 4 | **1,651.48** | **27.51** |
| Proposed (Same prob 0,3) | **12** | 1,706.18 | 28.13 |
| Proposed (Diff prob) | 4 | 1,727.70 | 30.72 |
| LP | Standard (0,3) | 0 | **235.75** | **6.85** |
| Proposed (Same prob 0,3) | 0 | 253.36 | 7.63 |
| Proposed (Diff prob) | 0 | 252.31 | 7.28 |
| CLR | Standard (0,3) | 7 | **3,078.49** | **151.07** |
| Proposed (Same prob 0,3) | 0 | 3,348.69 | 165.67 |
| Proposed (Diff prob) | **13** | 3,267.17 | 162.15 |

**Notes:** and **↓** showing the best performance

However, same with previous scenario that in term of computation efficiency, our proposed model still not outperform compare with standard dropout. If we look at the difference in training time, the MLC-AA with proposed method is not significantly less compare with the standard dropout. The training and testing time on MLC-AA with proposed method same prob vs diff prob also show not significantly less computation time.

### 4.1.3 Result on Scenario 3

The result on Experimental Scenario 3 was aims to find best probability dropout and do the experiment by hyperparameter tunning the Proposed Model vs Standard Dropout. In Table 4.5 and 4.6 show all result regarding hyperparameter tunning using Proposed Model vs Standard Dropout as follows:

Table 4.5 Comparison Result on Example based Evaluation of Hyperparameter Tunning for the Proposed Model vs. Standard Dropout

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Comparison Hyper Tuning** | **Test Params** | | | **Sub-set Accuracy** | **Hamming Loss↓** | **Accuracy (exp)** | **Precision (exp)** | **Recall (exp)** | **F1-Score (exp)** | **One error↓** | **Coverage↓** | **Ranking Loss↓** | **Average precision** |
| **Standard Dropout** | **0.0355** | | | **0.4091** | **0.1288** | **0.8364** | **0.9996** | **1.0094** | **0.9091** | **0.0694** | **6.2505** | **0.087** | **0.9293** |
| 0.0358 | | | 0.4096 | 0.1294 | 0.8364 | 0.9996 | 0.9879 | 0.9099 | 0.0717 | 6.2503 | 0.0871 | 0.9305 |
| 0.1362 | | | 0.4002 | 0.1332 | 0.8364 | 0.9996 | 1.0117 | 0.9 | 0.0722 | 6.2538 | 0.0876 | 0.9285 |
| 0.1772 | | | 0.4029 | 0.1321 | 0.8364 | 0.9996 | 1.0012 | 0.9057 | 0.0693 | 6.2555 | 0.0878 | 0.9286 |
| 0.2237 | | | 0.3995 | 0.1332 | 0.8364 | 0.9996 | 0.9785 | 0.9096 | 0.0699 | 6.2602 | 0.0885 | 0.9258 |
| 0.2299 | | | 0.3926 | 0.1348 | 0.8364 | 0.9996 | 0.994 | 0.9062 | 0.0694 | 6.2687 | 0.0899 | 0.9269 |
| 0.248 | | | 0.3994 | 0.1336 | 0.8364 | 0.9996 | 0.9946 | 0.9058 | 0.0695 | 6.261 | 0.0886 | 0.9247 |
| 0.2519 | | | 0.3787 | 0.1397 | 0.8364 | 0.9996 | 0.9757 | 0.9052 | 0.0705 | 6.2799 | 0.0917 | 0.9204 |
| 0.3269 | | | 0.3404 | 0.1635 | 0.8364 | 0.9996 | 0.9979 | 0.9023 | 0.0719 | 6.3181 | 0.0968 | 0.8365 |
| 0.3448 | | | 0.3831 | 0.1383 | 0.8364 | 0.9996 | 0.9799 | 0.9048 | 0.0717 | 6.2835 | 0.0923 | 0.922 |
| 0.3802 | | | 0.3567 | 0.1461 | 0.8364 | 0.9996 | 0.9875 | 0.9069 | 0.0704 | 6.2741 | 0.0908 | 0.9197 |
| **Proposed Dropout** | 0.0950 | 0.1241 | 0.4922 | 0.4022 | 0.1329 | 0.8364 | 0.9996 | 0.9917 | 0.9065 | 0.0698 | 6.2634 | 0.0890 | 0.9272 |
| 0.2642 | 0.2021 | 0.3016 | 0.4051 | 0.1317 | 0.8364 | 0.9996 | 0.9974 | 0.9076 | 0.0694 | 6.2584 | 0.0882 | 0.9278 |
| 0.2831 | 0.2673 | 0.2308 | 0.4021 | 0.1324 | 0.8364 | 0.9996 | 0.9880 | 0.9086 | 0.0692 | 6.2626 | 0.0887 | 0.9278 |
| 0.2965 | 0.2033 | 0.2617 | 0.4097 | 0.1307 | 0.8364 | 0.9996 | 0.9863 | 0.9094 | 0.0691 | 6.2628 | 0.0888 | 0.9295 |
| **0.2974** | **0.2311** | **0.2885** | **0.4077** | **0.1298** | **0.8364** | **0.9996** | **0.9918** | **0.9104** | **0.0698** | **6.2627** | **0.0887** | **0.9290** |
| 0.3025 | 0.2257 | 0.2603 | 0.3965 | 0.1337 | 0.8364 | 0.9996 | 1.0031 | 0.9042 | 0.0692 | 6.2673 | 0.0895 | 0.9273 |
| 0.3169 | 0.2005 | 0.2917 | 0.3966 | 0.1340 | 0.8364 | 0.9996 | 0.9822 | 0.9091 | 0.0702 | 6.2633 | 0.0890 | 0.9276 |
| 0.3566 | 0.2613 | 0.3177 | 0.3996 | 0.1329 | 0.8364 | 0.9996 | 0.9834 | 0.9095 | 0.0697 | 6.2662 | 0.0893 | 0.9270 |
| 0.3573 | 0.0514 | 0.4736 | 0.3998 | 0.1334 | 0.8364 | 0.9996 | 1.0027 | 0.9050 | 0.0696 | 6.2636 | 0.0889 | 0.9278 |
| 0.3901 | 0.1915 | 0.3324 | 0.3886 | 0.1360 | 0.8364 | 0.9996 | 0.9832 | 0.9080 | 0.0691 | 6.2630 | 0.0888 | 0.9252 |
| 0.4149 | 0.1911 | 0.4090 | 0.3903 | 0.1363 | 0.8364 | 0.9996 | 0.9823 | 0.9064 | 0.0695 | 6.2737 | 0.0905 | 0.9250 |
| 0.4195 | 0.4530 | 0.3511 | 0.3814 | 0.1382 | 0.8364 | 0.9996 | 0.9831 | 0.9071 | 0.0700 | 6.2786 | 0.0914 | 0.9215 |
| 0.4508 | 0.4953 | 0.4059 | 0.3817 | 0.1383 | 0.8364 | 0.9996 | 0.9861 | 0.9066 | 0.0700 | 6.2776 | 0.0912 | 0.9211 |
| 0.4840 | 0.1357 | 0.4780 | 0.3810 | 0.1388 | 0.8364 | 0.9996 | 0.9916 | 0.9057 | 0.0777 | 6.2762 | 0.0915 | 0.9215 |

**Notes:** and **↓** showing the best performance

Table 4.6 Comparison Result on Label based Evaluation of Hyperparameter Tunning for the Proposed Model vs. Standard Dropout

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Comparison Hyper Tuning** | **Test Params** | | | **Accuracy (micro)** | **Accuracy (macro)** | **Precision (micro)** | **Precision (macro)** | **Recall (micro)** | **Recall (macro)** | **F1-Score (micro)** | **F1-Score (macro)** | **AUC (micro)** | **AUC (macro)** |
| **Standard Dropout** | **0.0355** | | | **0.8712** | **0.8712** | **0.8946** | **0.8797** | **0.9592** | **0.9426** | **0.9257** | **0.9091** | **0.8667** | **0.7819** |
| 0.0358 | | | 0.8706 | 0.8706 | 0.8905 | 0.8747 | 0.9638 | 0.9485 | 0.9257 | 0.9099 | 0.8678 | 0.7824 |
| 0.1362 | | | 0.8668 | 0.8668 | 0.9026 | 0.8841 | 0.9424 | 0.9181 | 0.9221 | 0.9 | 0.8644 | 0.78 |
| 0.1772 | | | 0.8679 | 0.8679 | 0.8941 | 0.8773 | 0.9553 | 0.9367 | 0.9237 | 0.9057 | 0.8636 | 0.779 |
| 0.2237 | | | 0.8668 | 0.8668 | 0.8823 | 0.8677 | 0.9702 | 0.9576 | 0.9242 | 0.9096 | 0.8634 | 0.7753 |
| 0.2299 | | | 0.8652 | 0.8652 | 0.8869 | 0.8713 | 0.9614 | 0.9454 | 0.9227 | 0.9062 | 0.8606 | 0.7754 |
| 0.248 | | | 0.8664 | 0.8664 | 0.8903 | 0.8736 | 0.9585 | 0.9411 | 0.9231 | 0.9058 | 0.8608 | 0.7741 |
| 0.2519 | | | 0.8603 | 0.8603 | 0.878 | 0.8635 | 0.9674 | 0.9538 | 0.9205 | 0.9052 | 0.8594 | 0.767 |
| 0.3269 | | | 0.8365 | 0.8365 | 0.8365 | 0.8365 | 1 | 1 | 0.911 | 0.9023 | 0.7775 | 0.5 |
| 0.3448 | | | 0.8617 | 0.8617 | 0.8817 | 0.8659 | 0.9641 | 0.9489 | 0.921 | 0.9048 | 0.8588 | 0.7682 |
| 0.3802 | | | 0.8539 | 0.8539 | 0.8623 | 0.8573 | 0.9822 | 0.9756 | 0.9183 | 0.9069 | 0.8563 | 0.7639 |
| **Proposed Dropout** | 0.0950 | 0.1241 | 0.4922 | 0.8771 | 0.8771 | 0.9001 | 0.8834 | 0.9696 | 0.9525 | 0.9335 | 0.9165 | 0.8761 | 0.7887 |
| 0.2642 | 0.2021 | 0.3016 | 0.8783 | 0.8783 | 0.9008 | 0.8846 | 0.9703 | 0.9537 | 0.9343 | 0.9176 | 0.8733 | 0.7859 |
| 0.2831 | 0.2673 | 0.2308 | 0.8776 | 0.8776 | 0.8970 | 0.8820 | 0.9746 | 0.9600 | 0.9342 | 0.9186 | 0.8735 | 0.7863 |
| 0.2965 | 0.2033 | 0.2617 | 0.8793 | 0.8793 | 0.8986 | 0.8828 | 0.9746 | 0.9597 | 0.9351 | 0.9194 | 0.8745 | 0.7896 |
| **0.2974** | **0.2311** | **0.2885** | **0.8802** | **0.8802** | **0.8982** | **0.8827** | **0.9765** | **0.9623** | **0.9357** | **0.9204** | **0.8739** | **0.7921** |
| 0.3025 | 0.2257 | 0.2603 | 0.8763 | 0.8763 | 0.9038 | 0.8872 | 0.9635 | 0.9442 | 0.9327 | 0.9142 | 0.8717 | 0.7850 |
| 0.3169 | 0.2005 | 0.2917 | 0.8760 | 0.8760 | 0.8919 | 0.8775 | 0.9797 | 0.9671 | 0.9337 | 0.9191 | 0.8729 | 0.7863 |
| 0.3566 | 0.2613 | 0.3177 | 0.8771 | 0.8771 | 0.8935 | 0.8790 | 0.9788 | 0.9658 | 0.9342 | 0.9195 | 0.8719 | 0.7847 |
| 0.3573 | 0.0514 | 0.4736 | 0.8766 | 0.8766 | 0.9025 | 0.8855 | 0.9656 | 0.9471 | 0.9330 | 0.9150 | 0.8747 | 0.7877 |
| 0.3901 | 0.1915 | 0.3324 | 0.8740 | 0.8740 | 0.8900 | 0.8760 | 0.9796 | 0.9670 | 0.9326 | 0.9180 | 0.8708 | 0.7840 |
| 0.4149 | 0.1911 | 0.4090 | 0.8737 | 0.8737 | 0.8929 | 0.8773 | 0.9752 | 0.9605 | 0.9322 | 0.9164 | 0.8720 | 0.7832 |
| 0.4195 | 0.4530 | 0.3511 | 0.8718 | 0.8718 | 0.8873 | 0.8738 | 0.9805 | 0.9682 | 0.9315 | 0.9171 | 0.8690 | 0.7780 |
| 0.4508 | 0.4953, | 0.4059 | 0.8717 | 0.8717 | 0.8883 | 0.8744 | 0.9789 | 0.9660 | 0.9314 | 0.9166 | 0.8685 | 0.7782 |
| 0.4840 | 0.1357 | 0.4780 | 0.8712 | 0.8712 | 0.8891 | 0.8746 | 0.9771 | 0.9634 | 0.9310 | 0.9157 | 0.8680 | 0.7781 |

**Notes:** and **↓** showing the best performance

In this case, we do MLC-AA using the provided method for hyperparameter tweaking. The experiment utilized the Python module Optuna, an open-source framework for hyperparameter optimization in machine learning models [49]. The primary function is to automate the hyperparameter tuning process to identify the most optimal set of hyperparameter values for a given model. Hyperparameters are predefined configuration values for a model that are established before the training process and are not derived from the data. Optimizing these hyperparameters is essential for enhancing model performance. From the result in Table 4.5 and 4.6 are achieve based on the trials on experimental parameters as shown below:

Table 4.7 Hyperparameters Tuning Trials

|  |  |  |
| --- | --- | --- |
| **Hyperparameters Tuning Trials** | **Standard Dropout** | **Proposed Method** |
| Number of finished trials: | 50 | 50 |
| Number of pruned trials: | 39 | 36 |
| Number of complete trials: | 11 | 14 |
| Best trial (Subset-Accuracy): | 0.4091 | 0.4077 |
| Params: dropout\_rate: | 0.0355 | dropout\_rate1: 0.2974 |
| dropout\_rate2: 0.2311 |
| dropout\_rate3: 0.2885 |

The number of finished trials is total number of experiment for each hyperparameter. In this scenario we used dropout ratio for standard and proposed method 0.0 until 0.5. The pruned trials is the process of stopping or skipping trials that are unlikely to yield better results than the ones already completed that conduct by Hyperband algorithm [50]. The number of complete trial means that the experiment was completed achieved trained and evaluated using a specific set of hyperparameters as shown in Table 4.5 and 4.6. After a trial is completed, the result of objective function (subset-accuracy) or best trial is saved, and the hyperparameter configuration linked to that trial is treated as a data point in the search space. The best hyperparameters for standard dropout 0.0355 and the proposed method has three channel such as dropout\_rate1: 0.2974; dropout\_rate2: 0.2311; dropout\_rate3: 0.2885. From the result show that our proposed method can outperformed even the dropout ratio are optimized using hyperparameters tuning as describe in the following Table 4.8 below:

Table 4.8 Result evaluation and efficiency from optimum proposed Model Vs standard Dropout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MLC-Framework with DNN** | | **Sum of Metrics** | **Time Training (Second)** ↓ | **Time Testing (Second)** ↓ |
| AA | Standard Dropout | 6 | 431 | 3 |
| Proposed Method | 14 | 441 | 4 |

In this scenario, the number of metric of MLC-AA with proposed method is achieved significantly compare with standard dropout. MLC-AA with proposed method achieved number of metric 16 metric and the standard dropout achieved 6 metric. In this scenario, the result of efficiency on training and testing still not achieved significantly. It seem the architecture of our proposed method effect on performance. However, as we can see that the time training of our proposed method has difference only 10 seconds and the testing time only 1 seconds which is still can tolerated compare the achievement on sum of metrics.

## 4.2 Discussion

In this sub-section will discuss several key findings from the experimental result. Through rigorous analysis and comprehensive research, we have unearthed significant insights that shed light on various aspects of applying the proposed method in the context MLC, improvement our proposed method, effect number of channel dropout, effect dropout ratio, impact hyper tuning parameters and the last but not least is trade-off between evaluation on performance and time training and testing. These findings not only contribute to our understanding of our proposed method but also have broader implications for regularization of DNN model. By delving into these findings, we aim to provide a deeper understanding of regularization and its relevance in contemporary discourse. Moreover, we will explore the implications of these findings for future research directions and practical applications in regularization.

### 4.2.1 Finding on Applying Proposed Multi-Channel Weighted Dropout for Multi-Label Classification

From the result scenario 1 and 2, MLC-AA with proposed methods has been successfully achieving several performance of MLC evaluation. As mentioned in the result that the proposed method impact DNN to be more robust as a model classification for risk prediction diabetes complication. Furthermore, in the scenario 3, MLC-AA with proposed methods also outperformed compare to the standard dropout if we look at the number of metric evaluation of MLC in term of hyperparameter tuning. To explain the performance of the best method, the MLC-AA with proposed methods for risk prediction diabetic complication, we use AUC and ROC curves to show the trade-off between recall/sensitivity/true positive rate and false-positive rate, and the precision-recall curve. Figure 3 shows that the MLC-AA with proposed methods has the highest AUROC:

|  |
| --- |
|  |
| Figure 3. AUROC of MLC-AA with proposed methods |

Coronary heart disease (Y2) had the greatest AUROC of all diabetes complications, followed by Heart Attack (Y3), Stroke (Y4) and Arthritis (Y6). This data supported the MLC-AA with proposed methods AUC-macro value of 0. 0.7921.

However, our findings show that MLC-PT Label Powerset method in multi-label classification has several limitations. These include data sparsity, imbalanced class distributions, loss of label dependencies, and computational complexity. Data sparsity arises due to the exponential growth of class combinations, potentially hindering model generalization. Imbalanced class distributions impact model performance, favouring overrepresented classes. Label dependencies may be overlooked, affecting predictive accuracy. Additionally, the curse of dimensionality complicates model interpretation and increases computational demands. Alternative or hybrid approaches, tailored to dataset characteristics and task objectives, offer potential solutions to these challenges.

### 4.2.2 Improvements by Multi-Channel Weighted Dropout

Figure …plots the trends in validation accuracy and training accuracy against epoch rates. Moreover, the trends in validation loss and training loss against epoch rates also presented for two model; trained MLC-AA DNN with proposed method and trained with standard dropout. One of the key points of comparison is the disparity between training and validation accuracies. If the training accuracy is significantly higher than the validation accuracy, it indicates potential overfitting. Overfitting occurs when the model learns to fit the training data too closely, capturing noise and irrelevant patterns that do not generalize well to new, unseen data. Another crucial aspect to compare is the difference between training and validation loss. The training loss indicates the model's performance on the training data. The calculation involves the discrepancy between the model's predictions and the true targets (labels) in the training dataset. During training, the primary objective is usually to reduce this loss, as it reflects the model's ability to fit the training data. Validation loss indicates the model's ability to generalize to new data. The validation loss is computed in the same manner as the training loss, but it is based on a distinct validation dataset that was not used for training the model. The validation loss serves to offer an approximation of the model's performance on unfamiliar data.

|  |  |
| --- | --- |
| Validation Accuracy | Training Accuracy |
| A graph with red and blue lines  Description automatically generated | A graph with red and blue lines  Description automatically generated |
| Validation Loss | Training Loss |
| A graph with a line graph  Description automatically generated | A graph with a line graph  Description automatically generated |
| Figure 2. Trends accuracy and loss for validation and training set | |

Figure 2 illustrates how, for training and validation sets, our suggested approach performed better in terms of accuracy and loss than standard dropout. Since there is no significant difference in term accuracy or loss between the training and validation results, neither model exhibits any signs of overfitting. Following training, we discovered that the networks trained using our suggested technique had, on average, lower error rates than the traditional dropout.

### 4.2.3 Effects of parameters Multi-Channel Weighted Dropout on performance

From Figure… illustrate comparison accuracy or loss between the training and validation results in term of number of channel in our proposed method as follows:

|  |  |
| --- | --- |
| Validation Accuracy | Training Accuracy |
|  |  |
| Validation Loss | Training Loss |
|  |  |
| Figure 4.1 trends accuracy and loss for validation and training set on effect number of channel | |

For validation and training accuracy, during training, smaller number of channel achieved better accuracy, especially in channel 3. Meanwhile, the validation accuracy, if the higher number of channel achieved better accuracy. Particularly, in channel 32 that showing the higher number of validation accuracy. Furthermore, in term of validation and training loss, show that the higher number of channel achieved better loss value, especially the number of channel is 64 in validation and 32 in training.

### 4.2.4 Effects of parameter Dropout Probability Ratio on performance

The dropout ratio is a crucial parameter that regulates the proportion of neurons that are discarded.

|  |  |
| --- | --- |
| Validation Accuracy | Training Accuracy |
| A graph of different colored lines  Description automatically generated | A graph of different colored lines  Description automatically generated |
| Validation Loss | Training Loss |
| A graph of different colored lines  Description automatically generated | A graph of different colored lines  Description automatically generated |
| Figure | |

Dari Figure … dapat dilihat bahwa untuk training dan validation loss semakin kecil nilai dropout rationya semakin bagus error ratenya, dimana error rate terbaik dicapai saat dropout rationya 0,1. Dapat dilihat juga bahwa pada setiap nilai dropout, model kami menunjukkan error rate yang lebih bagus dibandingkan dengan standard dropout.

Untuk training dan validation accuracy, dapat dilihat bahwa semakin besar dropout rationya semakin bagus akurasinya, dimana akurasi tertinggi dicapai saat dropout rationya 0,9.

These results show that our proposed method does not depend on a specific dropout ratio to achieve improvements and that it can be used with a wide range of dropout ratio settings.

### 4.2.5 Trade-off between Performance and Time Training and Testing

Dari eksperimen yang telah dilakukan metode kami menunjukkan kinerja yang lebih baik daripada metode standar dropout dengan perbedaan waktu yang tidak signifikan. Hasil yang lebih baik menunjukkan kemampuan pendekatan alternatif dalam mengelola kompleksitas data dan model tanpa mengorbankan akurasi, sementara perbedaan waktu yang kecil menunjukkan efisiensi komputasi yang memadai.

Optimasi dari model kami juga dapat dicapai dengan lebih banyak melakukan tuning terhadap hyperparameter yang tersedia. Sehingga dapat mencapai efisiensi pada komputasi tanpa harus mengorbankan performa model.

* **Architecture Complexity Trade-off:**
  + Discuss the trade-off between model performance and computational efficiency.
  + Elaborate on how the simpler architecture of standard dropout contributes to better computational efficiency.
* **Potential Optimizations:**
  + Explore potential avenues for optimizing the proposed model to enhance its computational efficiency without compromising performance.
  + Discuss whether there are specific aspects of the architecture that can be streamlined or modified.

# 5. CONCLUSTION AND FURTHER RESEARCH

In this paper, we described multi-sample dropout, a regularization technique for accelerating training and improving generalization.

The key is creating multiple dropout samples at the dropout layer while the original dropout creates only one sample.

Multi-sample dropout can be easily implemented using existing deep learning frameworks by duplicating a part of the network after the dropout layer.

Experimental results using image classification tasks demonstrated that multi-sample dropout reduces training time and improves accuracy.

Because of its simplicity, the basic idea of the multi-sampling technique can be used in wide range of neural network applications and tasks.

# ACKNOWLEDGEMENT

# CRediT authorship contribution statement

Nur Rachman Dzakiyullah,

M. A. Burhanuddin,

Raja Rina Raja Ikram,

Novanto Yudistira,

Muhammad Rifqi Fauzi,

Novanto Yudistira: Conceptualization, Data curation, Formal

analysis, Methodology, Software, Visualization, Writing - original draft. Sutiman Bambang Sumitro: Conceptualization, Data

curation, Formal analysis, Supervision. Alberth Nahas: Writing

- original draft, Writing - review & editing, Validation. Nelly

Florida Riama: Data curation, Investigation, Writing - original

draft, Writing - review & editing, Validation.

# Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

# REFERENCES

[1] Y. Li *et al.*, “A Survey on Dropout Methods and Experimental Verification in Recommendation,” *IEEE Trans. Knowl. Data Eng.*, pp. 1–20, 2022, doi: 10.1109/TKDE.2022.3187013.

[2] B. Santra, A. Paul, and D. P. Mukherjee, “Deterministic dropout for deep neural networks using composite random forest,” *Pattern Recognit. Lett.*, vol. 131, pp. 205–212, 2020, doi: 10.1016/j.patrec.2019.12.023.

[3] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, “Improving neural networks by preventing co-adaptation of feature detectors,” pp. 1–18, Jul. 2012, [Online]. Available: http://arxiv.org/abs/1207.0580

[4] B. Lengerich, E. P. Xing, and R. Caruana, “On Dropout, Overfitting, and Interaction Effects in Deep Neural Networks,” in *arXiv*, 2020, pp. 1–17.

[5] S. Dong, P. Wang, and K. Abbas, “A survey on deep learning and its applications,” *Comput. Sci. Rev.*, vol. 40, p. 100379, May 2021, doi: 10.1016/j.cosrev.2021.100379.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[7] Z. Lu, C. Xu, B. Du, T. Ishida, L. Zhang, and M. Sugiyama, “LocalDrop: A Hybrid Regularization for Deep Neural Networks,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 7, pp. 1–1, 2021, doi: 10.1109/TPAMI.2021.3061463.

[8] H. Zheng, M. Chen, W. Liu, Z. Yang, and S. Liang, “Improving deep neural networks by using sparse dropout strategy,” *2014 IEEE China Summit Int. Conf. Signal Inf. Process. IEEE ChinaSIP 2014 - Proc.*, pp. 21–26, 2014, doi: 10.1109/ChinaSIP.2014.6889194.

[9] B. Li and X. Nong, “Automatically classifying non-functional requirements using deep neural network,” *Pattern Recognit.*, vol. 132, p. 108948, Dec. 2022, doi: 10.1016/j.patcog.2022.108948.

[10] X. Shen, X. Tian, T. Liu, F. Xu, and D. Tao, “Continuous Dropout,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 9, pp. 3926–3937, Sep. 2018, doi: 10.1109/TNNLS.2017.2750679.

[11] S. Hahn and H. Choi, “Understanding dropout as an optimization trick,” *Neurocomputing*, vol. 398, no. February, pp. 64–70, 2020, doi: 10.1016/j.neucom.2020.02.067.

[12] A. Iosifidis, A. Tefas, and I. Pitas, “DropELM: Fast neural network regularization with Dropout and DropConnect,” *Neurocomputing*, vol. 162, pp. 57–66, 2015, doi: 10.1016/j.neucom.2015.04.006.

[13] H. Salehinejad and S. Valaee, “EDropout: Energy-Based Dropout and Pruning of Deep Neural Networks,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 10, pp. 5279–5292, Oct. 2022, doi: 10.1109/TNNLS.2021.3069970.

[14] B. Ko, H. G. Kim, and H. J. Choi, “Controlled dropout: A different dropout for improving training speed on deep neural network,” *2017 IEEE Int. Conf. Syst. Man, Cybern. SMC 2017*, vol. 2017-Janua, pp. 972–977, 2017, doi: 10.1109/SMC.2017.8122736.

[15] G. Lee, H. Park, S. Ryu, and H. J. Lee, “Acceleration of DNN training regularization: Dropout accelerator,” *2020 Int. Conf. Electron. Information, Commun. ICEIC 2020*, pp. 2–3, 2020, doi: 10.1109/ICEIC49074.2020.9051194.

[16] C. Tingting, Z. Yang, X. Jianlin, and C. Huafeng, “Improved Convolutional Neural Network Fault Diagnosis Method Based on Dropout,” *Proc. - 2020 7th Int. Forum Electr. Eng. Autom. IFEEA 2020*, pp. 753–758, 2020, doi: 10.1109/IFEEA51475.2020.00160.

[17] Y. Gal, J. Hron, and A. Kendall, “Concrete Dropout,” *Neural Inf. Process. Syst.*, no. Nips 2017, May 2017, [Online]. Available: https://pdfs.semanticscholar.org/452a/c69a8af5366778f3be178966433f2b6bf70a.pdf

[18] S. I. Wang and C. D. Manning, “Fast dropout training,” *30th Int. Conf. Mach. Learn. ICML 2013*, vol. 28, no. PART 1, pp. 777–785, 2013.

[19] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” *33rd Int. Conf. Mach. Learn. ICML 2016*, vol. 3, pp. 1651–1660, 2016.

[20] H. Wang, W. Yang, Z. Zhao, T. Luo, J. Wang, and Y. Tang, “Rademacher dropout: An adaptive dropout for deep neural network via optimizing generalization gap,” *Neurocomputing*, vol. 357, pp. 177–187, Sep. 2019, doi: 10.1016/j.neucom.2019.05.008.

[21] P. Baldi and P. Sadowski, “The dropout learning algorithm,” *Artif. Intell.*, vol. 210, no. 1, pp. 78–122, May 2014, doi: 10.1016/j.artint.2014.02.004.

[22] A. Koivu, J. P. Kakko, S. Mäntyniemi, and M. Sairanen, “Quality of randomness and node dropout regularization for fitting neural networks,” *Expert Syst. Appl.*, vol. 207, no. June, p. 117938, 2022, doi: 10.1016/j.eswa.2022.117938.

[23] Y. Chen and Z. Yi, “Adaptive sparse dropout: Learning the certainty and uncertainty in deep neural networks,” *Neurocomputing*, vol. 450, pp. 354–361, Aug. 2021, doi: 10.1016/j.neucom.2021.04.047.

[24] Z. Liu, Z. Xu, J. Jin, Z. Shen, and T. Darrell, “Dropout Reduces Underfitting,” Mar. 2023, [Online]. Available: http://arxiv.org/abs/2303.01500

[25] Y. Mae, W. Kumagai, and T. Kanamori, “Uncertainty propagation for dropout-based Bayesian neural networks,” *Neural Networks*, vol. 144, pp. 394–406, 2021, doi: 10.1016/j.neunet.2021.09.005.

[26] X. Ying, “An Overview of Overfitting and its Solutions,” *J. Phys. Conf. Ser.*, vol. 1168, no. 2, 2019, doi: 10.1088/1742-6596/1168/2/022022.

[27] A. Labach, H. Salehinejad, and S. Valaee, “Survey of Dropout Methods for Deep Neural Networks,” Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.13310

[28] K. Sanjar, A. Rehman, A. Paul, and K. Jeonghong, “Weight dropout for preventing neural networks from overfitting,” *2020 8th Int. Conf. Orange Technol. ICOT 2020*, pp. 10–13, 2020, doi: 10.1109/ICOT51877.2020.9468799.

[29] J. Xie *et al.*, “Advanced Dropout: A Model-Free Methodology for Bayesian Dropout Optimization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 9, pp. 4605–4625, 2022, doi: 10.1109/TPAMI.2021.3083089.

[30] Q. Chen, W. Zhang, K. Zhu, D. Zhou, H. Dai, and Q. Wu, “A novel trilinear deep residual network with self-adaptive Dropout method for short-term load forecasting,” *Expert Syst. Appl.*, vol. 182, no. March, p. 115272, 2021, doi: 10.1016/j.eswa.2021.115272.

[31] S. Wang, T. Zhou, and J. A. Bilmes, “Jumpout: Improved dropout for deep neural networks with RELus,” *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 11565–11573, 2019.

[32] H. Salehinejad and S. Valaee, “Ising-Dropout: A Regularization Method for Training and Compression of Deep Neural Networks,” in *44th IEEE International Conference on Acoustics, Speech and Signal Processing (IEEE ICASSP), 2019*, Feb. 2019, pp. 3602–3606. doi: 10.48550/arXiv.1902.08673.

[33] S. Liu, Y. Gao, Y. Shen, M. Zhang, J. Li, and P. Sun, “Application of three statistical models for predicting the risk of diabetes,” *BMC Endocr. Disord.*, vol. 19, no. 1, p. 126, Dec. 2019, doi: 10.1186/s12902-019-0456-2.

[34] S. Kumari, D. Kumar, and M. Mittal, “An ensemble approach for classification and prediction of diabetes mellitus using soft voting classifier,” *Int. J. Cogn. Comput. Eng.*, vol. 2, pp. 40–46, Jun. 2021, doi: 10.1016/j.ijcce.2021.01.001.

[35] B. P. Nguyen *et al.*, “Predicting the onset of type 2 diabetes using wide and deep learning with electronic health records,” *Comput. Methods Programs Biomed.*, vol. 182, p. 105055, 2019, doi: 10.1016/j.cmpb.2019.105055.

[36] P. Pawara, E. Okafor, M. Groefsema, S. He, L. R. B. Schomaker, and M. A. Wiering, “One-vs-One classification for deep neural networks,” *Pattern Recognit.*, vol. 108, 2020, doi: 10.1016/j.patcog.2020.107528.

[37] H. Wu, S. Yang, Z. Huang, J. He, and X. Wang, “Type 2 diabetes mellitus prediction model based on data mining,” *Informatics Med. Unlocked*, vol. 10, no. August 2017, pp. 100–107, 2018, doi: 10.1016/j.imu.2017.12.006.

[38] Z. Xie, O. Nikolayeva, J. Luo, and D. Li, “Building Risk Prediction Models for Type 2 Diabetes Using Machine Learning Techniques,” *Prev. Chronic Dis.*, vol. 16, no. 9, p. 190109, Sep. 2019, doi: 10.5888/pcd16.190109.

[39] M. Rahman, D. Islam, R. J. Mukti, and I. Saha, “A deep learning approach based on convolutional LSTM for detecting diabetes,” *Comput. Biol. Chem.*, vol. 88, no. April, p. 107329, 2020, doi: 10.1016/j.compbiolchem.2020.107329.

[40] G. Swapna, R. Vinayakumar, and K. P. Soman, “Diabetes detection using deep learning algorithms,” *ICT Express*, vol. 4, no. 4, pp. 243–246, 2018, doi: 10.1016/j.icte.2018.10.005.

[41] H. Zhou, R. Myrzashova, and R. Zheng, “Diabetes prediction model based on an enhanced deep neural network,” *Eurasip J. Wirel. Commun. Netw.*, vol. 2020, no. 1, 2020, doi: 10.1186/s13638-020-01765-7.

[42] K. Kantawong, S. Tongphet, P. Bhrommalee, N. Rachata, and S. Pravesjit, “The Methodology for Diabetes Complications Prediction Model,” *2020 Jt. Int. Conf. Digit. Arts, Media Technol. with ECTI North. Sect. Conf. Electr. Electron. Comput. Telecommun. Eng. ECTI DAMT NCON 2020*, pp. 110–113, 2020, doi: 10.1109/ECTIDAMTNCON48261.2020.9090700.

[43] X. Xie, M. Xie, A. J. Moshayedi, and M. H. Noori Skandari, “A Hybrid Improved Neural Networks Algorithm Based on L2 and Dropout Regularization,” *Math. Probl. Eng.*, vol. 2022, pp. 1–19, 2022, doi: 10.1155/2022/8220453.

[44] CDC, “The Behavioral Risk Factor Surveillance System (BRFSS),” 2022. https://www.cdc.gov/brfss/ (accessed Oct. 29, 2022).

[45] S. Kaul and Y. Kumar, “Artificial Intelligence-based Learning Techniques for Diabetes Prediction: Challenges and Systematic Review,” *SN Comput. Sci.*, vol. 1, no. 6, pp. 1–7, 2020, doi: 10.1007/s42979-020-00337-2.

[46] M.-L. Zhang and Z.-H. Zhou, “A Review on Multi-Label Learning Algorithms,” *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 8, pp. 1819–1837, Aug. 2014, doi: 10.1109/TKDE.2013.39.

[47] L. Zhou, X. Zheng, D. Yang, Y. Wang, X. Bai, and X. Ye, “Application of multi-label classification models for the diagnosis of diabetic complications,” *BMC Med. Inform. Decis. Mak.*, vol. 21, no. 1, pp. 1–10, 2021, doi: 10.1186/s12911-021-01525-7.

[48] J. Bogatinovski, L. Todorovski, S. Džeroski, and D. Kocev, “Comprehensive comparative study of multi-label classification methods,” *Expert Syst. Appl.*, vol. 203, no. February, 2022, doi: 10.1016/j.eswa.2022.117215.

[49] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, “Optuna,” in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, in KDD ’19. New York, NY, USA: ACM, Jul. 2019, pp. 2623–2631. doi: 10.1145/3292500.3330701.

[50] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, “Hyperband: A novel bandit-based approach to hyperparameter optimization,” *J. Mach. Learn. Res.*, vol. 18, pp. 1–52, 2018.

# Appendix 1

Table 1. factors used in this research based behavioral factors during 2016-2021 [44]

|  |  |  |  |
| --- | --- | --- | --- |
| No | Factors | Question | Value |
| 1 | GENHLTH | General Health | 1: Excellent, 2: Very good, 3: Good, 4: Fair, 5: Poor |
| 2 | MENTHLTH | Number of Days Mental Health Not Good | 1: 0–10, 2: 11–20, 3: 21-30 |
| 3 | HLTHPLN1 | Have any health care coverage | 1: Yes, 2: No |
| 4 | CHECKUP | Length of time since last routine check-up | 1: <1 y, 2: 1–2 y, 3: 3–5 y, 4: >5 y, 6: Never |
| 5 | EXERANY | Exercise in Past 30 Days | 1: Yes, 2: No |
| 6 | SMOKER | Computed Smoking Status | 1: Current smoker every day, 2: Current smoker some days, 3: Former smoker, 4: Never smoked |
| 7 | TOTINDA | Leisure Time Physical Activity Calculated Variable | 1: Had physical activity or exercise, 2: No physical activity in past 30 days |
| 8 | RFDRHV | Heavy Alcohol Consumption Calculated Variable | 1: No; 2: Yes |
| 9 | MARITAL | Marital Status | 1: Married, 2: Divorced, 3: Widowed, 4: Separated, 5: Never married, 6: Unmarried couple |
| 10 | RENTHOM1 | Own or Rent Home | 1: Own, 2: Rent, 3: Other |
| 11 | EMPLOY | Employment Status | 1: Employed, 2: Self-employed, 3: No work >1 y, 4: No work <1 y, 5: Homemaker, 6: Student, 7: Retired, 8: Unable to work |
| 12 | INCOME2 | Income Level | 1: <$10 K, 2: $10–$15 K, 3: $15–$20 K, 4: $20–$25 K, 5: $25–$35 K, 6: $35–$50 K, 7: $50–$75 K, 8: >$75 K |
| 13 | EDUCAG | Computed level of education completed categories | 1: Did not graduate high school, 2: Graduated high school, 3: Attended college, 4: Graduated college |
| 14 | MSCODE | Metropolitan Status Code | 1: Centre city, 2: County, 3: Suburban, 5: not in MSA |
| 15 | BLIND | Blind or Difficulty seeing | 1: Yes, 2: No |
| 16 | DECIDE | Difficulty Concentrating or Remembering | 1: Yes, 2: No |
| 17 | SEX | Calculated sex variable | 1: Male, 2: Female |
| 18 | FLSHOT | Flu Shot Calculated Variable | 1: Yes, 2: No |
| 19 | RACE | Computed Race-Ethnicity grouping | 1: White, 2: Black, 3: American Indian or Alaskan Native, 4: Asian, 5: Native Hawaiian or other Pacific Islander, 6: Other race, 7: Multiracial, 8: Hispanic |
| 20 | BMICAT | Computed body mass index categories | 1: Underweight, 2: Normal weight, 3: Overweight, 4: Obese |
| 21 | AGEGYR | Reported age in five-year age categories calculated variable | 1: 31 to 40 y, 2: 41–50 y, 3: 51–60 y, 4: 61–70 y, 5: 71–80 y, 6: >81 y |
| 22 | DEAF | Are you deaf or do you have serious difficulty hearing? | 1: Yes, 2: No |
| 23 | DIABETE | (Ever told) you have diabetes | 1: Yes, 2: Yes but pregnant, 3: No, 4: Prediabetes |
| 24 | PDIABTST | Had a test for high blood sugar or diabetes in the past three years? | 1: Yes; 2: Yes, during pregnancy; 3: No |
| 25 | PREDIAB1 | Ever been told by a doctor or other health professional that you have pre-diabetes or borderline diabetes? | 1: Yes; 2: Yes, during pregnancy; 3: No |
| 26 | RFBMI | Overweight or obese calculated variable | 1: No (Notes: 1200 <= \_BMI5 < 2500 (BMI5 has 2 implied decimal places); 2: Yes (Notes: 2500 <= \_BMI5 < 9999) |
| 27 | CHCKDNY | (Ever told) you have kidney disease? | 1: Yes, 2: No |
| 28 | CVDCRHD | Ever Diagnosed with Angina or Coronary Heart Disease | 1: Yes, 2: No |
| 29 | CVDINFR | Ever Diagnosed with Heart Attack | 1: Yes, 2: No |
| 30 | CVDSTRK | Ever Diagnosed with a Stroke | 1: Yes, 2: No |
| 31 | CHCOCNC | (Ever told) you had any other types of cancer? | 1: Yes, 2: No |
| 32 | HAVARTH | Told Have Arthritis | 1: Yes, 2: No |
| 33 | ADDEPEV | Ever told you had a depressive disorder | 1: Yes, 2: No |

# Appendix 2

Table 4. Model Evaluation Metrics for MLC

|  |  |  |
| --- | --- | --- |
| **Example Based Metric** | | |
| **Subsets Accuracy** | |  |
| **Hamming Loss** | |  |
| **Accuracy (exp)** | |  |
| **Precision (exp)** | |  |
| **Recall (exp)** | |  |
| **F1-Score (exp)** | |  |
| **One error** | |  |
| **Coverage** | |  |
| **Ranking Loss** | |  |
| **Average precision** | |  |
| **Label-Based Metric** | | |
| **Accuracy (micro)** |  | |
| **Accuracy (macro)** |  | |
| **Precision (micro)** |  | |
| **Precision (macro)** |  | |
| **Recall (micro)** |  | |
| **Recall (macro)** |  | |
| **F1-Score (micro)** |  | |
| **F1-Score (macro)** |  | |
| **AUC (micro)** |  | |
| **AUC (macro)** |  | |