DaeYoung Kim[1]

[1]Affiliation not available

April 18, 2024

Abstract

Under the commonalities found in the goals of two areas, neural network compression and feature selection for dimension reduction, this research focused on finding a new method to address both issues: a method that can lead to easier feature selection, and an enhancement in the capacity of information flow control of neural network compression techniques, especially clustering based compression. Specifically, this research focused on creating a novel and effective framework to transform the weight matrix between the input layer and the first hidden layer in neural networks to be optimal. In other words, a method that can make the weight matrix's structure itself optimal for information extraction. By proposing a simple, yet powerful weight clipping + GMM based method called an In-and-Out Weight Box that can intrinsically act similar to filtering while increasing the possibility of getting better results in compression, the main aim of research was found to be satisfied. Using Glioma Grading data from the UCI Repository for checking performance of the In-and-Out Weight box in fitting neural networks, it was found that significantly better compression results can be achieved in terms of weight sharing via clustering. This research also suggests a new feature selection method based on the In-and-Out Weight box constraint called IOW-FI, which can lead to solving limitations or problems of filtering techniques such as setting the number of components to be selected as efficient features or considering joint distributions of feature space.

In-and-Out Weight Box: A novel approach for better network compression and feature selection

*Daeyoung Kim

***Abstract*— Under the commonalities found in the goals of two areas, neural network compression and feature selection for dimension reduction, this research focused on finding a new method to address both issues: a method that can lead to easier feature selection, and an enhancement in the capacity of information flow control of neural network compression techniques, especially clustering based compression. Specifically, this research focused on creating a novel and effective framework to transform the weight matrix between the input layer and the first hidden layer in neural networks to be optimal. In other words, a method that can make the weight matrix's structure itself optimal for information extraction. By proposing a simple, yet powerful weight clipping + GMM based method called an In-and-Out Weight Box that can intrinsically act similar to filtering while increasing the possibility of getting better results in compression, the main aim of research was found to be satisfied. Using Glioma Grading data from the UCI Repository for checking performance of the In-and-Out Weight box in fitting neural networks, it was found that significantly better compression results can be achieved in terms of weight sharing via clustering. This research also suggests a new feature selection method based on the In-and-Out Weight box constraint called IOW-FI, which can lead to solving limitations or problems of filtering techniques such as setting the number of components to be selected as efficient features or considering joint distributions of feature space.**

***Index Terms*—Feature Selection, Filtering, Gaussian Mixture Models, Information Flow, MCMC, Neural Network Compression, Weight Clipping, Weight Clustering, Weight Constraint.**

This article is a preprint submitted to TechRxiv, powered by IEEE. Submission Date: March 31, 2024 (GMT).

Daeyoung, Kim. is a undergraduate student in Yonsei University, Department of Applied Statistics, Seoul, Korea (Corresponding author and First Author) (e-mail: lgtlsafg@yonsei.ac.kr).

Data(Glioma Grading Clinical and Mutation Features) used in this research is based on UCI Repository. Check the link below for further explanations (https://archive.ics.uci.edu/dataset/759/glioma+grading+clinical+and+mutation+features+dataset).

The overall process of this research used Python's scikit-learn mixture package, tensorflow package and PyMC(PyMC3) package. Brief explanations of main procedures in Appendix.

I. INTRODUCTION

When regarding the architectures and natures of recently used neural network compression techniques, such as weight clustering or pruning, it is clear that the ultimate goal of most weight compressions for supervised-deep learning is to decrease total memory usage for storing a trained model with preserving performance. However, there seems to be another 'unintended' goal which is undermined when considering the areas of lightening neural network: information flow control optimization. Making models less overfit and robust to data shifts, it seems that main algorithms of network compression techniques not only result in preserving performance, but sometimes also result in finding better information flow structures than the original model or finding the optimal information flow architecture which filters out irrelevant information from inputs or feature sets, while preserving important feature information for accurate predictions on response variables[1][2][3].

These characteristics of network compression highly overlap with the main task most feature dimension reduction studies focus on. Consider, for example, one sub-area: feature selection. Filtering techniques such as Mutual Information Feature Selection(MIFS, MIFS-U) or minimum redundancy maximum relevance (mRMR) algorithms try to find the best subset of input vectors using iterative approaches with sequentially selecting k number of covariates that can maximize the mutual information between explanatory variables and the response variable y[4][5]. Meanwhile, wrapping techniques such as Harris Hawk Optimization (HHO) algorithms or recently introduced algorithms such as Salp Swarm Algorithms(SSA) based on Swarm Intelligence or Particle Swarm Optimization (PSO) focus on finding the best subset of input vectors that maximize the performance of a given evaluation function while searching a binarized variable plane using meta-heuristic approaches based on non-gradient, nature oriented hunting assumptions[6][7]. Both feature selection techniques, filtering and wrapping, focus on classifying input variables based on the relevance with response variable y, which, in a broad sense, is an area of finding the optimal information flow control: degrading the information of irrelevant covariates, while emphasizing the information from important covariate sets.

Under the commonalities found in goals of two areas, compression and feature selection, this research focused on finding a new method to address both issues: a method that

can lead to easier feature selection, and an enhancement in the capacity of information flow control of neural network compression techniques, especially layer wise weight clustering. Focusing on the first weight matrix(let us denoted it as W1) between input layer and the first hidden layer, this research assumed that information in W1 has a dominant ability to supply information in both directions of the neural network. For backwards, fitted W1's absolute sum or rank of weight values per input variable node can be an evaluation criteria for each input feature, whereas in forwards, fitted W1 can work as the most powerful component when controlling information flows, which directly affect the performance of neural network prediction and network compression results. This implies that optimizing W1 and extracting maximum information from W1 with validity is a key to conduct high-performing deep learning architectures.

This research focused on creating a novel and effective framework to transform W1 to be optimal, specifically, a method that can make W1's structure itself optimal for information extraction. By proposing a simple, yet powerful weight clipping based method called an In-and-Out Weight Box that can intrinsically act similar to filtering while increasing the possibility of getting better results in compression, the main aim of research was found to be satisfied. Throughout construction and applications of the In-and-Out Weight box, this study ultimately aims to practically alleviate difficulties for finding an optimal subset of input information or optimal compressed network structures when fitting a valid supervised-deep learning model.

II. Preliminaries

A. Gaussian Mixture Models(GMM)

Gaussian Mixture Model (GMM) is a probabilistic distribution-based approach used in fields where data can be clustered or classified to two or more groups. Using a mixture of normal distributions, the GMM architecture is constructed as in **(1)**.

$$[GMM\ Architecture(K\ clusters)]$$

$$\pi_i: i^{th}\ groups'\ weight\ parameter, \sum_{i=1}^{K} \pi_i = 1$$

$$\gamma_i(x) = \frac{\pi_i f_i(x;\mu_i,\Sigma_i)}{\Sigma_j \pi_j f_j(x;\mu_j,\Sigma_j)} = p(z=i\,|x),\ \ f_i \sim N(\mu_i,\Sigma_i)$$

$$P(X) = \sum_{i=1}^{K} \pi_i \times Normal(X;\mu_i,\Sigma_i)\ ,\boldsymbol{\theta} =< \mu,\pi,\Sigma >$$

$$EM\ Algorithm\ \Longrightarrow find\ \boldsymbol{\theta}\ that\ maximize \log(P(\boldsymbol{X})) \quad \textbf{(1)}$$

With π_i, μ_i and Σ_i each denoting the prior probability (weight, mixture coefficient) of i^{th} group, mean of i^{th} group and the covariance of i^{th} group, GMM fits the model by using Expectation Maximization (EM) algorithm. Through running the iterative process of EM algorithms, parameters compromising the distribution of each group are updated until convergence. With the result of GMMs, one can compute the allocation probability function of a group ($\gamma_i(x)$= p(Z = 'i' | x), Z: latent variable which denotes clusters) when some data x is given, which can lead to analysis or predictions on classification possibilities for individual samples. Due to its flexibility and high performance in extracting existing clusters from data, currently GMM is used not only in fields for classification analysis but also in anomaly detection areas. By connecting with other deep learning architectures such as Generative Adversarial Networks (GAN), Autoencoders or Long-Short-Term-Memory(LSTM) cells, GMM is currently contributing in improving the ability to find anomalies in massive big data[8][9].

B. Compression: weight clustering based compression

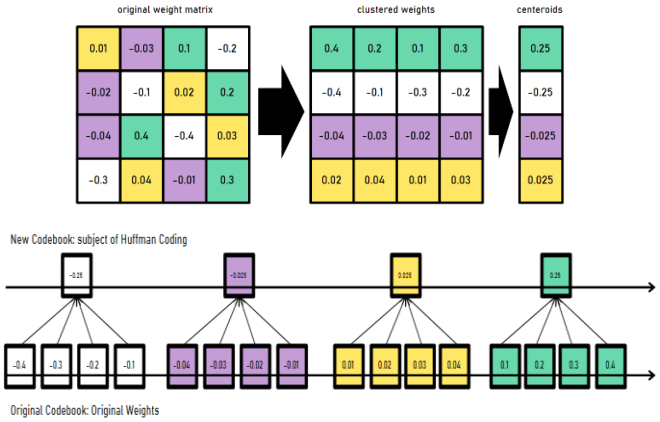

Fig. 1. Weight clustering example on 4 x 4 weight matrix

Along with pruning techniques, weight clustering is a popular network compression method in various research fields. Weight clustering is a neural network compression technique which starts from the idea of sharing weights to have lower memory cost while preserving information in the original structure[10]. In compression via weight clustering, each value in the weight matrix between layers is considered as a sample. These samples are then clustered by algorithms such as k-means clustering, which leads to the process of weight allocation to generated groups(clusters). By using the centroids of each group(cluster), methods such as weight quantization transforms individual weights to the centroid values based on previous allocation. Thus, the overall complexity in the original weight matrix is decreased, while resulting dimension reduction in the codebooks for neural networks by using methods such as Huffman Coding. **Figure 1** is an example of weight clustering. By using the centroids of 0.25, -0.25, 0.025 and -0.025 based on clustering, the 4 by 4 initial weight matrix is reduced to only four quantities in the final codebook.

III. Methodology

A. Process of the In-and-Out Weight Box

The In-and-Out Weight Box is a margin creator that uses weight clipping under various conditioning. **Figure 2** briefly summarizes the main process of In-and-Out weight box application. Based on the spread of trained weights between the input layer and the first hidden layer, this method finds the optimal clipping values(boundaries) to make a certain margin which can lead to clearer compression of neural networks with minimization of information from input features that are irrelevant in predicting response variable y.

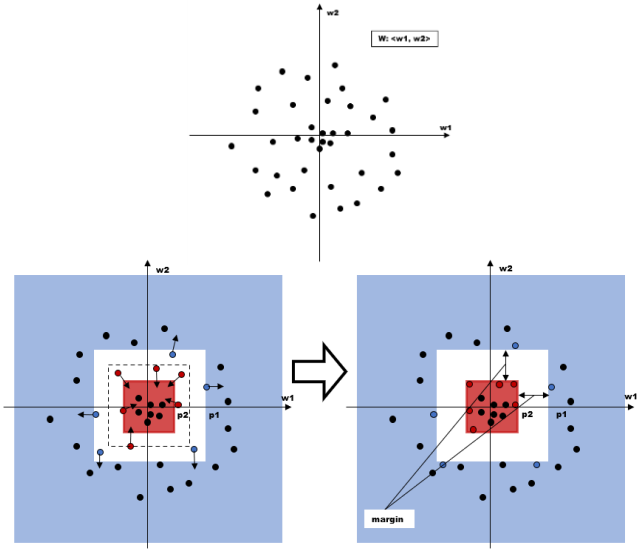

Fig. 2. The In-and-Out Weight Box constraint process assuming weight vector with two components. The first row is an example of original weight space W(= <w1, w2>) The second row in this figure shows the process of the original weight space being clipped by two in and out boxes creating a margin space to divide informative and non-informative weights.

This process starts by lightly fitting a pre-defined neural network on training data. After light fitting through methods such as early stopping algorithms, the first weight matrix(will denote as 'W1') linking the input layer and the first hidden layer is analyzed using GMMs. As weights in W1 control the information flows regarding the importance of each input features, finding weight clusters in W1 can not only result in extracting the importance of each edge but also practically result in classifying input features based on relevance.

When applying GMM analysis, the focus is not on the directionality of the effects of features but on the importance each input feature has. Therefore, absolute values of weights in W1 are used to generate clusters while ignoring signs of each weight. The number of clusters is set as two: one for effective weights which are far from 0(group 1) and another for ineffective weights close to 0(group 2). After convergence on the fitting procedure of GMM is guaranteed, this research uses location probabilities for each cluster to find appropriate clipping values(boundaries: p1, p2) that construct the In-and-Out Weight Box.

$$p_1 = {\gamma_1}^{-1}(\beta), p_2 = {\gamma_2}^{-1}(\alpha), \;\; \alpha, \beta > 0.5$$

$$\gamma_{1(2)}: group\ 1(2)'s\ \ location\ probability\ basd\ on\ GMM$$
$$\beta: hyper\ parameter\ close\ to\ 50\%$$
$$\alpha: hyper\ parameter\ close\ to\ 100\% \qquad \textbf{(2)}$$

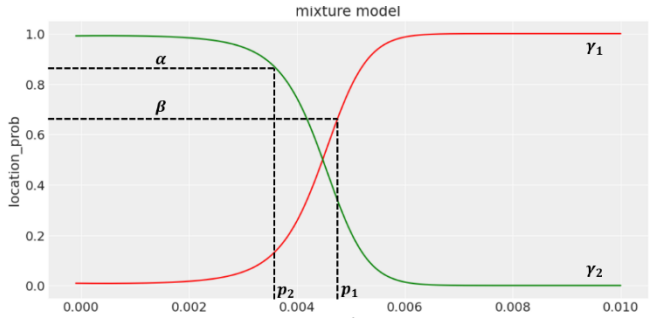

Fig. 3. Visualization of finding appropriate p1 and p2 by equation **(1)**

Let p1 and p2 (p1 > p2) each denote clipping values for the outer and inner weight box depicted in **Figure 2**. In the In-and-Out Weight Box process, p1 is used as a boundary to gather and extract effective weights found in fitted W1, whereas p2 is used to gather ineffective weights in W1. To filter out only highly unimportant information flows in W1, p2 should be a small value that guarantees substantially high location probability for group 2 under a GMM approach. On the other hand, to extract effective information flows as much as possible, p1 should be some low value while preserving the location probability for group 1 to be greater than probability for group 2. This can be rewritten in a mathematical form as **(2)**, which can be visualized as in **Figure 3**.

In this research, the problem of setting optimal p1 and p2 was considered solvable using two different thresholds(alpha, beta) for boundaries. For example, an alpha of 80%, and a beta of 60% can be a candidate for a feasible solution when finding appropriate values of p1 and p2 that satisfy the conditions above. After setting valid values for p1 and p2, the main process of the In-and-Out Weight Box is computed: the weight clipping procedure **(Algorithm 1)**. For absolute weight values below p2 or over p1, weights are preserved, as an absolute weight value below p2 can be considered as an 'almost definite' non-informative weight, while absolute value over p1 can be considered as an 'almost definite' high-informative weight based on the results of GMM analysis. For absolute values between p1 and p2, weights are clipped based on the euclidean distance between p1 and p2. This leads to a clipping procedure based on arithmetic mean of p1 and p2. If absolute

weight value is under (p1+p2)/2, it is clipped to p2 or -p2, whereas values over (p1+p2)/2 are clipped to p1 or -p1. As a result, a margin space with a width of (p1-p2) is created in the weight dimension.

Creating a void in weight space based on clipping controversial weights, the In-and-Out Weight Box can reinforce non-significant input features to have smaller impacts on the predicted response value, which can lead to enhancements in original network performance and compression results due to a better starting. Furthermore, in an opposite directional approach, this clipping procedure can lead to better feature selection. As important features will have higher sum of absolute weights in W1, whereas unimportant features will have lower sum of absolute weights in W1, the gap between each feature importance in terms of weights would have larger gaps, which can lead to higher confidence on selecting stopping points for filtering techniques or higher possibilities of directly finding an optimal feature subset in terms of feature selection. After construction of the In-and-Out Weight Box, the box is then applied as a weight constraint condition only for the weight matrix between input layer and first hidden layer in the original structure of the network. This way, by re-fitting with the box constraint under identical network structure, it is possible to optimize information flows when fitting deep learning models with the same train data on hand.

Algorithm 1 In-and-Out Weight Box constraint

[STEP 1] Find p1 and p2 by fitting GMM

I. Lightly fit a deep neural network with more than one hidden layer.
Extract W1 from the fitted neural network

II. Fit GMM with number of clusters = 2
Input: absolute weights from weight matrix W1
Output: Estimates of $\pi_{1(2)}$, $f_{1(2)} \sim N(\mu_{1(2)}, {\sigma_{1(2)}}^2)$
Compute location probability $\gamma_{1(2)}$

III. Find p1 and p2 using Newtons-method
Set parameters $\alpha, \beta, \varepsilon$(threshold)
Define $g_1(x) = \gamma_1(x) - \beta,\ g_2(x) = \gamma_2(x) - \alpha$

While $\varepsilon < |g_1(x) - 0|$:
  $x_{n+1} \leftarrow x_n - \frac{g_1(x_n)}{{g_1}'(x_n)}$
  if $\varepsilon \geq |g_1(x_{n+1}) - 0|$:
    Set p1 = x_{n+1} ; break
return p1

While $\varepsilon < |g_2(x) - 0|$:
  $x_{n+1} \leftarrow x_n - \frac{g_2(x_n)}{{g_2}'(x_n)}$
  if $\varepsilon \geq |g_2(x_{n+1}) - 0|$:
    Set p2 = x_{n+1} ; break
return p2

[STEP 2] Create the In-and-Out Weight Box

I . Create the In-and-Out Weight box constraint
For ∀ weights(w) in W1:

if |w| > (p1+p2)/2:
  if sign(w) == positive:
    clip_value(min_value= p1, max_value= K)
  else:
    clip_value(min_value= -K, max_value= -p1)
  (K: sufficiently large value that can replace ∞)
else:
  clip_value(min_value= -p2, max_value = p2)

return transformed weights

II. Transform and apply the above constraint as a tensorflow subclass constraint using the function:
tensorflow keras. constraints. Constraint.

This research used Python's scikit-learn package to compute GMM analysis. Specifically, the function of GaussianMixture in sklearn.mixture was implemented with maximum iteration set as 100 under random initial point setting. For fitting and checking convergence of the model, .fit( ) function and .converged_( ) function in GaussianMixture were used. Meanwhile, finding values p1 and p2 that satisfy two different thresholds was computed using newtons-method, and the In-and-Out Weight box constraint was built based on using tensorflow keras.constraints.Constraint instance and tf.clip_by_value function. Check the appendix for more detailed explanation about constraint-related codes.

B. Experiments for evaluation

To check performance, practical implementation of the In-and-Out Weight Box was computed. By comparing performance of a weight box based network with the originally fitted neural network and checking the existence of improvements in network compression when using a weight box constraint were executed. For training data and test data, 'Glioma Grading Clinical and Mutation Features' dataset from the UCI Machine Learning Repository was implemented. Glioma Grading data is an open access dataset based on TCGA-LGG and TCGA-GBM brain glioma projects, which contains basic information of glioma patients such as current glioma grade level(binarized target), isocitrate dehydrogenase mutation status, neurofibromin type 1 status and epidemal growth factors. Based on 23 covariates(3 clinical features and 20 genes) and one target variable, this dataset aims to find an optimal subset of mutation genes and clinical features for better prediction of glioma levels with low cost[11].

This research divided the total Glioma dataset into a proportion of 80(training data) : 20(test data). The former was used to train a simple neural network structure comprised of three hidden layers with 30, 20, 10 nodes sequentially, and an

output layer of one node with the activation function set as sigmoid. Activation functions for each hidden layer was set as ReLU, with Glorot weight initialization based on normal distribution and l2-kernel regularization attached to every hidden layer[12]. Under a full batch gradient descent update method, the original baseline neural network used binary cross entropy loss function and an Adam compiler with learning rate of 0.005 for fitting. Finally, for light training, a total of 500 epochs with an early stopping algorithm(patience=200) based on loss value was used, and the fitting result of each epoch was checked via an accuracy metric. The prediction accuracy of the original model was checked and compared with other variants of the model by test data after fitting. (Each components or structures of the model was constructed using Python tensorflow.keras tools in a jupyter notebook environment). After light fitting, the process of building the In-and-Out Weight box and its application to the original model was implemented (hyperparameter alpha, beta and threshold value set as 80%, 60% and 5*1e-2, with value K set as 1e+4). Using test data, the final step of comparing the performance of box-constraint based network with the original network and comparing the performance between compressed networks from each model was driven(As this research focused on improving the results of clustering based weight compression, weight clustering algorithms based on functions in tensorflow_model_optimization package were implemented with number of clusters set as 20 for high compression (initial spacing method was set as LINEAR [5])). Moreover, after significance of using the In-and-Out Weight box being checked, a novel feature selection method using the results of the weight box was suggested for better dimension reductions of input feature space. The procedure of the suggested feature selection(FS) is as follows (Algorithm 2):

Algorithm 2 In-and-Out Weight Box based FS

[STEP 1] Compute IOW-FI statistic

I. Calculate IOW-FI statistic values for each feature (M: number of nodes in first hidden layer.)

$$IOW-FI(X_i) \equiv (1-\lambda)\psi_i - \lambda\phi_i\,,\ \ (w_{ij} \in W1)$$
$$\psi_i = \frac{\sum_{j=1}^{M} I\{|w_{ij}| \geq p1\}}{M}\,,\ \ \phi_i = \frac{\sum_{j=1}^{M} I\{|w_{ij}| \leq p2\}}{M}$$
$$\lambda: weighting\ parameter$$

II. Re-arrange the order of features according to the results of sorting IOW-FI stat in descending order.

[STEP 2] Find optimal feature set S

I. Using Bayesian switch point modelling and MCMC sampling, find filtering point k that divides values of sorted IOW-FI stats.

$$[Switch\ Point(=\tau)\ \ modelling\ assuming\ Normal\ dist]$$
$$X \sim f(x_t) = \begin{cases} N(\mu_1, {\sigma_1}^2) & (t < \tau) \\ N(\mu_2, {\sigma_2}^2) & (t \geq \tau) \end{cases}$$
$$Find\ optimal\ \theta = <\mu, \sigma, \tau> by\ MCMC\ in\ PyMC$$

II. Filter out relatively unimportant features by only selecting k features before switch point.

III. Define selected k features as set S.

Return S; **End of algorithm**

By introducing a new statistic called 'In-and-Out Weight based Feature Importance'(IOW-FI), which uses absolute weights over p1 as an importance factor while using values under p2 as a degrading factor with interpolation parameter λ, and using bayesian switch point modelling using Python's PyMC package, this research aims to make filtering techniques more convenient in finding optimal values for setting k: the number of selected features, while indirectly addressing the importance of each feature under the joint distribution of feature space. Though some feature filtering techniques such as JMIM use joint distributions of features in iterative feature selection, many mutual information-based filtering methods (for example, NMIFS, mRMR, MIFS-U) rely on the distributions based on only the currently searched individual feature X and response value Y[13]. In this research, IOW-FI is expected to work as a solution to this limitation. Checking improvements in test accuracy when using IOW-FI on the original model based on W1 from the fitted box-constrained model, this paper ultimately aims to validify the effects of using In-and-Out Weight Box for better FS.(To check the use of PyMC codes, see Appendix).

IV. Experimental Results

A. In-and-Out Weight Box Construction for Glioma Data

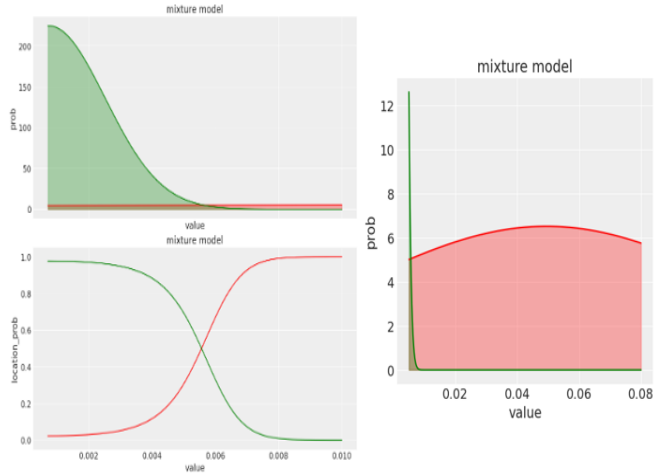

Fig. 4. Results in GMM analysis on W1 of the original baseline DNN. Figures on the left side show the pdf(green) of group 2 and the allocation probability graph of $\gamma_{1(2)}(x)$ in

interval [0.0007, 0.01]. Figure in right is a focused version of the pdf(red) of group 1 by setting the x interval as [0.005, 0.08]. When visualizing the fitted GMM, clear difference in the centers of each group can be recognized.

After lightly fitting the original baseline deep neural network depicted in III.B, using the absolute values of fitted W1, the GMM fitting was converged as in **Figure 4** and **Table I**. Visualization and parameter fitting results show that group 1 has a substantially large centroid value compared to group 2, whereas group 2's values are highly focused to the center compared to group 1. This implies that separation of weights based on relevance with the response variable(Glioma level) was successful in the aspect of distance and density. The crossover point for allocation probabilities of groups were found in interval [0.004, 0.006]. Thus, initial points for running newtons-method to find p1 and p2 were set as the arithmetic mean of 0.005. Under threshold of ε=0.05, values p1 and p2 were computed as 0.00600964 and 0.00462994 with newtons-method, which led to a margin box space with length of 0.0013797. Based on these values, the In-and-Out Weight Box constraint was constructed as in III.A.

TABLE I
GMM FITTING RESULTS: ESTIMATES FOR EACH GROUP

Results	π	μ	Σ
Group 1	0.53	0.0494	3.749e-03
Group 2	0.47	0.0007	3.159e-06

B. Model performance comparison based on test data

TABLE II
MODEL PERFORMANCE ON GLIOMA DATA

	OM	IOWB	OM-C	IOWB-C
Train data (accuracy)	**0.8838**	**0.8838**	0.8644	**0.8793**
Train data (Loss)	0.3465	**0.3456**	**0.3463**	0.3474
Test data (accuracy)	0.6726	**0.7143**	0.5536	**0.6964**
Test data (Loss)	0.8047	**0.7371**	0.9409	**0.7687**

*OM: Original Model, IOWB: In-and-Out Weight Box constraint based Model, OM-C: Compressed OM, IOWB-C: Compressed IOWB. Best results written in bold type(Epochs, Learning rate were all equally conditioned).

After finding values p1, p2 and constructing the In-and-Out Weight Box, comparison of models OM, IOWB, OM-C and IOWB-C were implemented to check performance of the box constraint. Numerical results based on accuracy and binary cross entropy loss were computed as in TABLE II. Results show the IOWB model outperforms OM in the aspect of overfitting issues or robustness in data shifts. While having identical or better results in training, IOWB model's accuracy

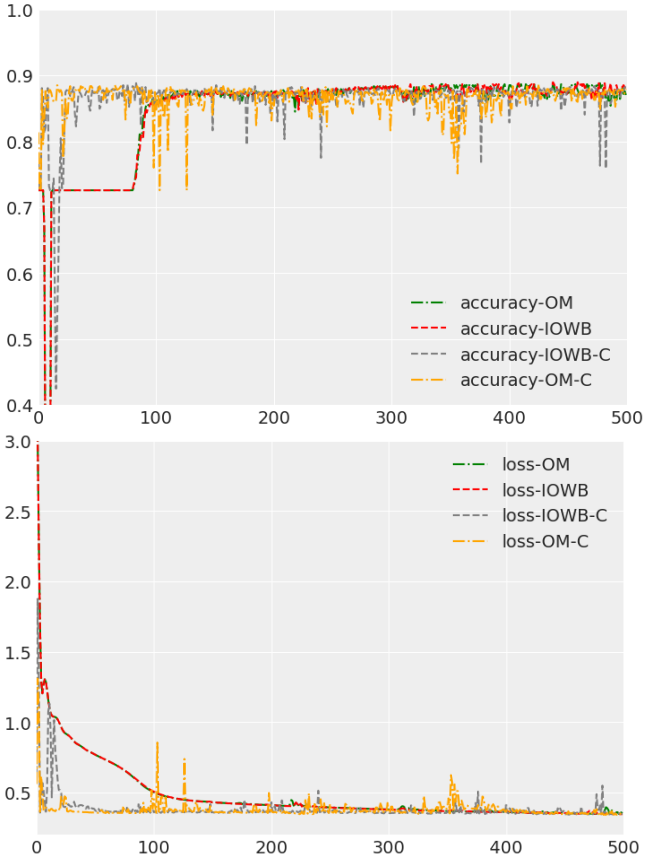

Fig. 5. Visualization of the training process(epochs=500) of four models: In all models, convergence in accuracy metric and binary cross entropy loss was found.

dropped 0.1695 for test data while OM's accuracy dropped 0.2112. This implies using the In-and-Out Weight box constraint itself is a better way of constructing baseline neural networks for further enhancements. The advantages of using the box constraint is more vivid when checking the results of compressed networks via clustering. Though same compression described in III.B was implemented, there was significant difference in the loss of performance between OM-C and IOWB-C. While the accuracy of OM-C on training data dropped from 0.8838 to 0.8644, the accuracy of IOWB dropped from the same value of 0.8838 to 0.8793. This implies when using the box-constraint, the amount of information loss by compression is less than using the original model, which can validify the expectation that the margin of boxes will work as a mean to provide clearer guidance for clustering based compression techniques.

This characteristic is more amplified in analyzing non-observed data: test data. In terms of test data, for the OM-C model, there has been a dramatic decrease in accuracy after compression(from train data to test data: 0.8644 to 0.5536, test data performance difference after compression: 0.6726 to 0.5536). On the other hand, IOWB-C model had minor reduction in accuracy(from train data to test data: 0.8793 to

0.6964, test data performance difference after compression: 0.7143 to 0.6964). Furthermore, when considering the performance in terms of loss value for test data, the gaps between the compressed model and the original model were substantially different(OM-C: 0.1362(0.8047 to 0.9409), IO WB-C: 0.0316 (0.7371 to 0.7687)). Thus, overall results deduced using Glioma data support the initial thought of this research that making W1 optimal with the In-and-Out Weight Box leads to the construction of better information flow architecture in supervised-deep learning. In other words, former expectations of the IOWB to enhance original network performance and network compression results were found to be satisfied.

C. Suggestion of a Novel FS method: IOW-FI

Based on the significance of using the In-and-Out Weight box, this research suggested a novel feature selection method using the results of In-and-Out Weight Box: the IOW-FI algorithm. Using IOWB's W1 as sample data, IOW-FI stat was calculated for each feature with λ set as 0.5 for balance. After computation, **Algorithm 2**'s Step2 was implemented under 5000 iterative sampling and 1000 tuning(per chain) based on the Metropolis-Hastings Markov Chain Monte Carlo(MCMC) sampler of PyMC (used parallel computing for two chains). Uniform distributions and exponential distributions were used as priors for $\mu_{1(2)}$ and $\sigma_{1(2)}$, each(*values for prior were selected based on descriptive statistics of IOW-FI stats such as min, max, standard deviations). Results of the MCMC were visualized as in **Figure 6** and **Figure 7**.(*Used python's arviz package).

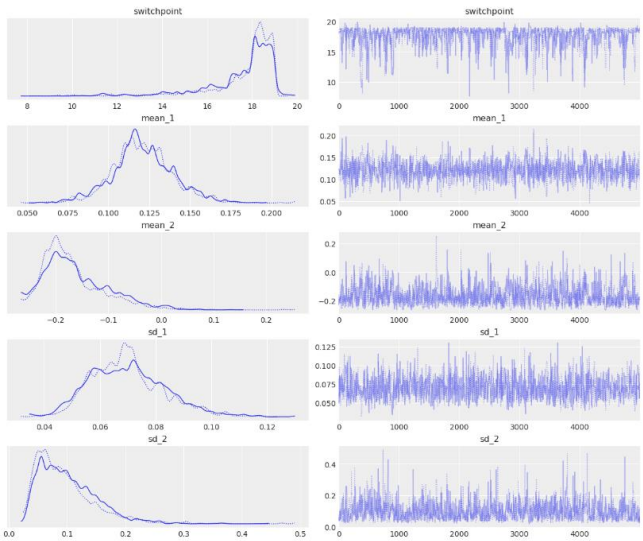

Fig. 6. Visualization of MCMC results for Glioma Data-based IOW-FI stats values. From top to bottom, $\tau, \mu_1, \mu_2, \sigma_1, \sigma_2$.

	mean	sd	hdi_3%	hdi_97%	mcse_mean	mcse_sd	ess_bulk	ess_tail	r_hat
switchpoint	17.639	1.534	14.478	19.007	0.075	0.053	463.0	681.0	1.00
mean_1	0.121	0.019	0.085	0.158	0.001	0.000	867.0	770.0	1.00
mean_2	-0.165	0.062	-0.266	-0.055	0.003	0.002	546.0	921.0	1.00
sd_1	0.070	0.013	0.047	0.097	0.000	0.000	849.0	1089.0	1.01
sd_2	0.100	0.053	0.028	0.190	0.002	0.001	564.0	1379.0	1.00

Fig. 7. Summary of MCMC for each parameters: From top to bottom, $\tau, \mu_1, \mu_2, \sigma_1, \sigma_2$'s summary statistic. As $\hat{R}$ values are all below 1.05, convergence of MCMC chains is checked.

When comparing the posterior means of each $\mu_{1(2)}$, significant difference between two estimates implies classification based on switch point could be valid. Based on this conclusion, this research used the posterior mean value of switch point MCMC samples for finding the optimal estimate for $\tau(\hat{\tau}=18)$. Therefore, by selecting 18 features based on the descending order of IOW-FI stat values, features 'Age_at_diagnosis', 'IDH1', 'NOTCH1', 'IDH2', 'PTEN', 'EGFR', 'GRIN2A', 'RB1', 'CIC', 'MUC16', 'CSMD3', 'PIK3R1', 'NF1', 'PIK3CA', 'TP53', 'BCOR', 'SMARCA4', 'FUBP1' were selected as important feature set S with IOW-FI values of 0.26666667, 0.2, 0.2, 0.16666667, 0.16666667, 0.16666667, 0.13333333, 0.13333333, 0.13333333, 0.1, 0.1, 0.1, 0.06666667, 0.06666667, 0.06666667, 0.06666667, 0.03333333, 0.03333333, sequentially.

Based on set S, this research fitted a new model with identical structures of the original baseline neural network. (Same parameters such as learning rates, weight initialization methods or kernel constraints were adopted). Performance results compared to OM and IOWB were deduced as in **Table III**. Using 78% of given features, the IOW-FI FS method

TABLE III
PERFORMANCE OF IOW-FI BASED MODEL

Results	OM	IOWB	IOW-FI
Train Data (accuracy)	**0.8838**	**0.8838**	0.8748
Train Data (Loss)	0.3465	**0.3456**	0.3693
Test Data (accuracy)	0.6726	**0.7143**	**0.7143**
Test Dat (Loss)	0.8047	**0.7371**	0.7707

*IOW-FI refers to the model fitted based on IOW-FI feature selection and MCMC. Best values in bold type.

succeeded in building a better model than the original model: OM, which used the entire feature set. Though performance in training data is slightly lower than the former model, for test data, IOW-FI shows substantially high performance than OM in terms of prediction accuracy and cross entropy loss while reducing the total dimension of input features. This result implies that the In-and-Out Weight Box not only contributes in enhancing network compression via clustering, but also can contribute in finding better feature subsets in the aspect of feature selection, which leads to the fact that expectations in

Introduction, which stated the possibilities of the In-and-Out Weight Box working as means to alleviate difficulties for finding an optimal subset of input information or optimal compressed network structures when fitting a valid supervised-deep learning model, are found valid.

Apart from the comparison between OM and IOW-FI, when comparing IOWB and IOW-FI, it is found that IOWB works better in predicting both train and test data. This implies that the In-and-Out Weight Box constraint's ability can expand to the area of replacing FS procedures, which can lead to large savings in computational cost.

V. Conclusion

This research suggested a novel approach called an In-and-Out Weight Box constraint which can enhance network compression results and alleviate difficulties for finding an optimal subset of input features. Results show that the new constraint can work as a mean to address these goals by making the model more robust and optimizing weight matrix W1 for better information extractions. Although there are limitations in considering methods to optimize hyper - parameters such as α, β, λ, it is expected that this novel constraint can work as means to enhance both network compression and feature dimension reduction in terms of providing efficient starting points or baselines. For further research, comparing IOW-SI with other feature selection techniques and improving the IOW-SI statistic in the aspects of finding optimal lambda or reducing information redundancy for advancements are currently planned. Please check briefly explained code procedures of the IOWB and **Algorithm 2** in Appendix.

Appendix

*Below are partial codes used in this paper.

Please check documents of tensorflow and pymc.io for thorough explanations of packages or functions. Especially, switch point modelling codes using pymc were based on the skeletons introduced in the presentation of Chris Fonnesbeck for PyData's London 2019 Conference.

(Brief Explanations)

1 . In-and-Out Weight Box Constraint Python code

#p1, p2: values from GMM analysis and Newtons-method.

```
p1=tf.constant(0.00600964)
p2=tf.constant(0.00462994) #classification boundaries computed by GMM
class in_out_box(tf.keras.constraints.Constraint):
    def __call__(self, w):
        return tf.where(condition=tf.greater(tf.abs(w),(p1+p2)*0.5),
                        x= tf.where(condition=tf.greater(w,0),
                                    x=tf.clip_by_value(w,clip_value_min= p1, clip_value_max=10000),
                                    y=tf.clip_by_value(w,clip_value_min=-10000,clip_value_max= (-1)*p1) ),
                        y= tf.clip_by_value(w,clip_value_min= (-1)*p2, clip_value_max=p2))
```

2. IOW-FI algorithm Python codes

***W_new : Matrix W1 of fitted IOWB network.**

(Re-shaped to 23 x 30)

```
IOW_FI = []
lam = 0.5
M = 30
for i in range(0,23,1):
    stt = (1-lam)*len(W_new[i,][np.abs(W_new[i,])>= 0.00600964])/M - lam*len(W_new[i,][np.abs(W_new[i,])<= 0.00462994])/M
    IOW_FI.append(stt)
np.argsort(IOW_FI)[::-1]

sort_IOW_FI = np.sort(IOW_FI)[::-1]

d_ = np.arange(len(sort_IOW_FI))
with pymc3.Model() as _model:
    switchpoint = Uniform('switchpoint', lower=d_.min(), upper=d_.max())
    mean_1 = Uniform('mean_1',np.min(sort_IOW_FI),np.max(sort_IOW_FI))
    mean_2 = Uniform('mean_2',np.min(sort_IOW_FI),np.max(sort_IOW_FI))
    sd_1 = Exponential('sd_1',lam=1/np.mean(np.std(sort_IOW_FI)))
    sd_2 = Exponential('sd_2',lam=1/np.mean(np.std(sort_IOW_FI)))
    rate1 = switch(switchpoint > d_, mean_1, mean_2)
    rate2 = switch(switchpoint > d_, sd_1, sd_2)
with _model:
    mi = Normal('mi', mu=rate1,sd=rate2,observed=sort_IOW_FI)
    starting = find_MAP()
with _model:
    samples = sample(5000, tune=1000, step=Metropolis(),cores=2, random_seed=1111, start=starting)

_val = int(np.round(np.mean(samples['switchpoint'])))

idx = np.argsort(IOW_FI)[::-1] [:_val]
S = X_tr.columns[idx];S
```

REFERENCES

[1] S. Han, J. Pool, J. Tran and W. J. Dally, “Learning both Weights and Connections for Efficient Neural Networks”, arXiv:1506.02626, 2015.

[2] N. Abuhajar, T. Sun, Z. Wang, S. Gong, C. D. Smith, X. Wang, L. Xu and J. Liu, “Network Compression and Frame Stitching for Efficient and Robust Speech Enhancement”, *IEEE National Aerospace and Electronics Conference*, pp. 269-276, 2021.

[3] T. Wu, X. Li, D. Zhou, N. Li and J. Shi, “Differential Evolution Based Layer-Wise Weight Pruning for Compressing Deep Neural Networks”, *SENSORS*, Vol. 21, no. 3, pp. 880-899, 2021.

[4] M. Bennasar, R. Sethchi and H. Yulia, “Feature Interaction Maximisation”, *Pattern Recognition Letters*, Vol. 34, no. 14, pp. 1630-1635, 2013.

[5] H. Alshamlan, G. Badr and Y. Alohali, “mRMR-ABC: A Hybrid Gene Selection Algorithm for Cancer Classification Using Microarray Gene Expression Profiling”, *BioMed Research International*, 2015, pp. 1-15, 2015.

[6] T. Thaher, and N. Arman, "Efficient Multi-Swarm Binary Harris Hawks Optimization as a Feature Selection Approach for Software Fault Prediction", *2020 11th International Conference on Information and Communication Systems*, 2020.

[7] M. Mahapatra, S. K. Majhi, S. K. Dhal, “MRMR-SSA: a hybrid approach for optimal feature selection”, *Evolutionary Intelligence*, Vol. 15, no. 3, pp. 2017-2036, 2022.

[8] V. Tra, M. Amayri and N. Bouguila, "Outlier detection via multiclass deep autodencoding Gaussian mixture model for building chiller diagnosis", *Energy and buildings*, Vol. 259, 2022.

[9] L. Jing, L. Pengbo, L. Huijun and C. Wanghu, “Outlier Detection Based on Stacked Autoencoder and Gaussian Mixture Model”, *2022 IEEE International Conference on Big Data*, 2022.

[10] S. Han, H. Mao and W. J. Dally, “Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding”, arXiv:1510.00149v5, 2016.

[11] E. Tasci, Y. Zhuge, H. Kaur, K. Camphausen and A. Krauze, "Hierarchical Voting-Based Feature Selection and Ensemble Learning Model Scheme for Glioma Grading with Clinical and Molecular Characteristics", *International Journal of Molecular Sciences*, Vol. 23, no. 22, pp. 14155, 2022.

[12] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks”, *Journal of Machine Learning Research*, Vol. 9, pp. 249-256, 2010.

[13] M. Bennasar, Y. Hicks and R. Setchi, “Feature selection using Joint Mutual Information Maximisation”, *Expert Systems with Applications*, Vol. 42, no. 22, pp. 8520-8532, 2015.