Purposeful Regularization with Reinforcement Learning for Facial Expression Recognition In-the-Wild

SangHwa Hong
Department of Industrial Engineering
Seoul National University of Science and Technology
Gongreung-ro 232, Nowon-gu, Seoul, South Korea
hongsw5911@seoultech.ac.kr

Abstract

Facial Expression Recognition (FER), an essential aspect of emotion analysis through artificial intelligence, is a crucial research area. Although traditional approaches utilizing Convolutional Neural Networks (CNNs) for analyzing human emotions from facial expressions achieve superior accuracy over conventional machine learning methods, overfitting —especially arising severely from data collected in uncontrolled, In-the-wild settings —significantly impedes CNNs performance. This is due to the data scarcity and inherent noise inside In-the-wild data. To address this challenge, this paper introduces a novel regularization method that employs Reinforcement Learning (RL) to adaptively apply regularization hyperparameters appropriate for the evolving state of trained CNNs. Through experiments on various datasets such as CIFAR100, FER2013, and AffectNet datasets including diverse perspective analysis such as graphical, Grad-CAM and numerical analysis, it is demonstrated that the suggested method can alleviate memorization of noise in training data and promote learning of essential features. The significance of the suggested method lies in its demonstrated remarkable effectiveness in enhancing CNNs' generalization and reducing overfitting.

1. Introduction

Facial Expression Recognition (FER) [22] is a significant area of research in emotion analysis through artificial intelligence. Facial expressions are a primary means for humans to convey emotions, and it is anticipated that AI systems will soon be adept at discerning and replicating human emotional states from these expressions during their interactions and communications with people. The practical applications of FER are vast, spanning sectors such as healthcare and entertainment, where it has the potential to significantly enhance the user experience.

Existing research indicates that leveraging Convolutional Neural Networks (CNNs) [2],[22] allows for a more accurate analysis of human emotions based on facial expressions than what is achievable with conventional machine learning approaches. Conventionally, research on FER has been conducted for the facial images obtained in controlled settings, such as laboratory environments with explicit emotional prompts [22]. Recently, most of the research has focused on In-the-wild facial expressions captured in uncontrolled environments for applications in practice [22].

In the domain of In-the-wild FER, due to the scarcity of large datasets and expression-unrelated variations referred to as noise [22], [40], [21], overfitting [40] becomes a significant issue for the practical application of CNNs. Overfitting, exacerbated by limited training data and noise, occurs when the CNNs memorize too much irrelevant noise instead of learning essential features from the training data and severely becomes dependent on the noise for the prediction, as described in Figure 1.

To distinguish the essential feature from noise, common approaches such as simplifying the model's architecture, using an ensemble and applying regularization techniques are utilized [11]. The regularization introduces additional constraints or penalties [20], [33] to the CNNs being trained, thereby, decreasing the generalization error on unseen data [43], [14]. Regularization techniques, universally recognized as effective, involve penalizing parameter magnitudes [20], injecting noise [27], and applying dropout [35]. Generally, hyperparameters of regularization are maintained statically or adjusted iteratively by a hand-crafted scheduler during training CNNs [30], [36]; refer to Diagram 1 and Diagram 2 in Figure 2. A key limitation of these approaches is their inability to adjust hyperparameters appropriate for the evolving state of the trained CNNs (state can be parameters of trained CNNs at a certain point which are updated by Gradient Descent [3]); contrast Diagram 1 and Diagram 2 against Diagram 3 in Figure 2.

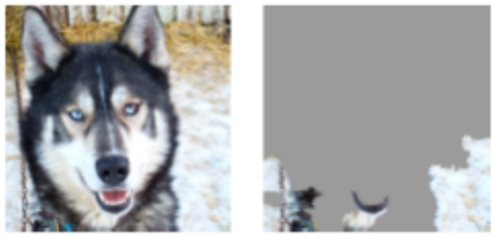

Figure 1. This figure shows the deep learning model's prediction for the class of data severely depends on the memorized noise in the training dataset, not the relevant representative features.

On the other hand, as described in Diagram 3 in Figure 2, adaptively navigating hyperparameters according to the dynamically evolving **state** during training CNNs can be more effective than disregarding the state of the trained CNNs in terms of distinguishing the essential features from noise [44], [5], [29], [42]. This approach observes the state of the trained CNNs, selects appropriate regularization hyperparameters based on observations, and then trains the CNNs under the chosen regularization hyperparameters.

Concretely, how regularization hyperparameters are chosen is decided by how the **objective** of updating hyperparameters is defined. The selection of hyperparameters is made in a direction that aims to achieve the defined objective. For example, in the representative approach where the objective of adjusting hyperparameters is defined as minimizing training loss, the hyperparameters are updated in a direction that effectively reduces the training loss. However, the defined objective of adjusting regularization hyperparameters must be distinguished from the objective of updating the parameters of trained CNNs, which aims to minimize training loss; refer to Diagram 3 in Figure 2.

Especially, by adopting the approach described in Diagram 3 of Figure 2 in the domain of In-the-Wild FER, the effectiveness of regularization can be significantly enhanced. As a precedent example following the approach described in Diagram 3 of Figure 2 for this domain, the research [13] utilizes the backpropagation-based optimization not only to update parameters of CNNs but also to update the regularization hyperparameters appropriate for state of trained CNNs [23], [6]. In this method, adjustments of both trained parameters and updated hyperparameters share the common objective of minimizing training loss. This means that the hyperparameters are also updated in a direction that can reduce the training loss by the Gradient Descent, which can be expressed by the Equation 1 where the θ_{hyper}, α and $L(\theta)$ are defined as hyperparameters value for regularization technique, learning rate and training loss generated by the forward propagation with whole trainable parameters. Through experiments, as evidenced by [13] and shown in Table 1, this approach has been proven to be distinctively effective in learning essential features. However, given that the goal of updating hyperparameters is to minimize training loss, this objective inherently leads CNNs to the memorization of noise within the training dataset, which in turn establishes a fundamental limitation in preventing the trained CNNs from memorizing this noise.

$$\theta_{hyper} := \theta_{hyper} - \alpha \cdot \frac{\partial L(\theta)}{\partial \theta_{hyper}} \tag{1}$$

To tackle overfitting in the In-the-Wild FER domain, we propose the novel method also adopting the approach in Diagram 3 of Figure 2 but overcoming the limitation of the approach 1. To overcome the limitation, the proposed method defines the objective of updating the regularization hyperparameters as preventing the trained CNNs from memorizing the noise in the training dataset while facilitating the CNNs to learn essential features, instead of minimizing the training loss. To implement this, proposed method leverages Reinforcement Learning (RL) [15], [38] to train the agent [1], [4], [32], which adjusts regularization hyperparameters (action of agent) applied for training of CNNs, in a direction that can effectively reduce the noise memorized by trained CNNs and facilitate the learning of essential features (objective of agent). Given that the RL algorithm optimizes the agent to conduct actions that can maximize the accumulated reward, by defining the MDP (consisting of state, action and reward [28], [37], [12], [31] and necessary for applying the RL optimization) which can lead the action to prevent the trained CNNs from memorizing the noise, the challenging overfitting issue can be alleviated distinctively. The MDP is presented in the method section.

As a result, the proposed method is proven to be distinctively effective for preventing the CNNs from memorizing the noise in the training data while learning the essential features.

Contributions: The proposed method contributes to the community facing challenges for applying FER applications in practice due to overfitting. It also fills the research gap of devising regularization techniques remarkably effective for In-the-wild FER, a domain where developing regularization still has not received sufficient academic attention.

Rest of papers: In the related work, the concepts related to RL and concept of Bayesian Optimization are presented. Because the suggested method is implemented by the RL algorithm, the concepts related to the RL are presented. Moreover, since Bayesian Optimization has both similar and different points compared to the suggested method, it can be useful for understanding the suggested method. In the method section, the MDP formulation is suggested. In experiments, diverse types of analysis show that the suggested method is effective for In-the-Wild FER domain by distinctively alleviating the overfitting.

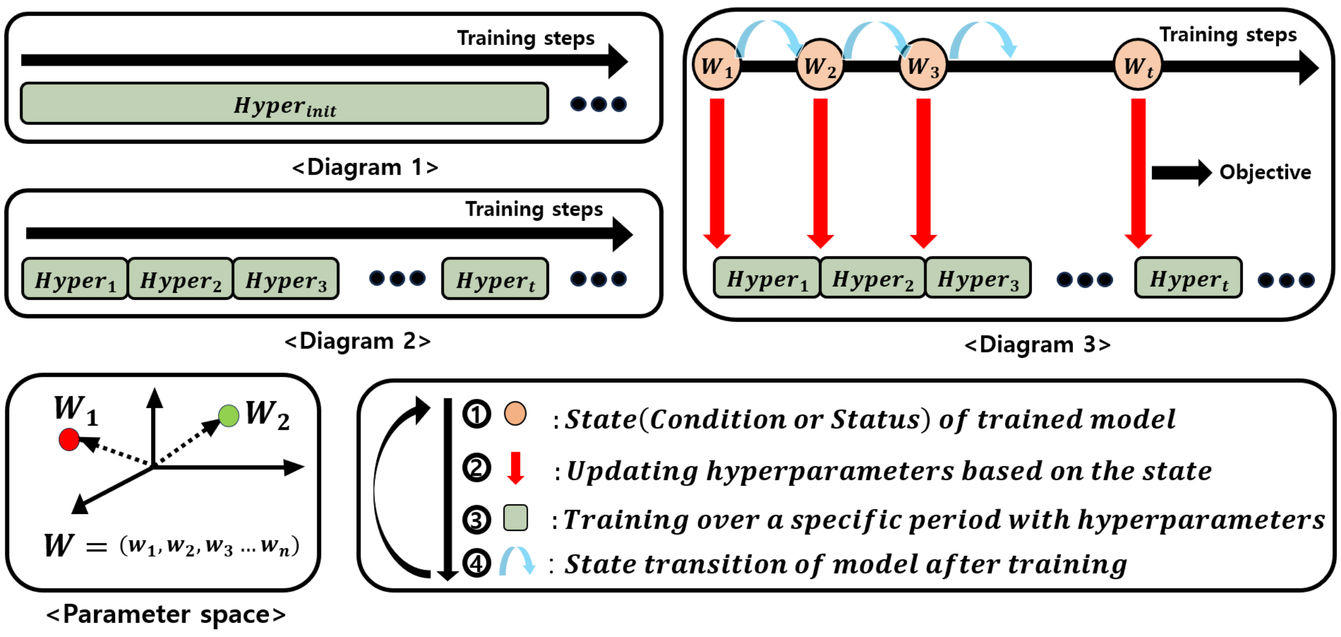
Training steps
$Hyper_{init}$
<Diagram 1>
Training steps
$Hyper_1$ $Hyper_2$ $Hyper_3$ $Hyper_t$
<Diagram 2>
W_1 W_2 W_3 W_t
Training steps
Objective
$Hyper_1$ $Hyper_2$ $Hyper_3$ $Hyper_t$
<Diagram 3>
W_1 W_2
$W = (w_1, w_2, w_3 \ldots w_n)$
<Parameter space>
① : State(Condition or Status) of trained model
② : Updating hyperparameters based on the state
③ : Training over a specific period with hyperparameters
④ : State transition of model after training

Figure 2. This figure shows three types of hyperparameter adjustment strategies during training the CNNs. $Hyper_t$ and W_t are the applied hyperparameter during training process and parameters of the trained CNNs for $t = 1, 2, \ldots t$. The parameter space [34] is the space where each dimension represents a different parameter that can be adjusted during the training process. In this figure, the state is defined as the parameters of CNNs at certain point which are updated by the Gradient Descent.

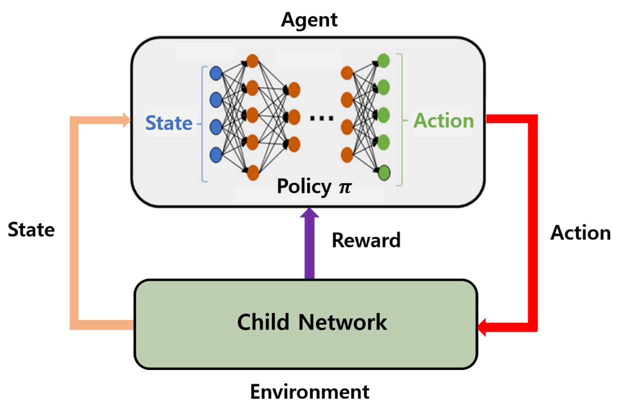

Figure 3. This figure shows the relationship among State, Action, Reward, Agent, Policy, Environment and Child Network.

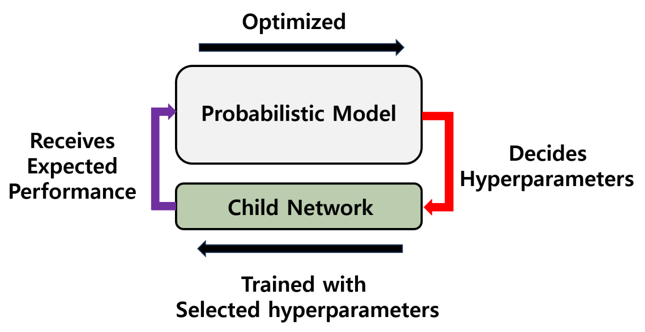
Optimized
Probabilistic Model
Receives Expected Performance
Decides Hyperparameters
Child Network
Trained with Selected hyperparameters

Figure 4. This figure shows how the Bayesian Optimization works for Hyperparameter Optimization

2. Related Work

As mentioned in the Rest of papers, this section presents the concepts of RL related concepts and Bayesian Optimization.

2.1. MDP, Environment and RL

Utilizing the RL algorithm requires formulating the Markov Decision Process (MDP) consisting of State, Action, and Reward [28], [37], [12], [31]; refer to Figure 3. In a Markov Decision Process (MDP), a state represents the specific condition or status at a given time, which is considered for deciding the action. An action is a choice made by the policy network of agent [1], [4]. The reward [9], [18], [8], [17], [26] is the quantitative assessment on the effectiveness of action.

Under the defined MDP, the agent takes action on the constructed Environment, thereby, receiving the reward for the corresponding action and the transitioned state; refer to Figure 3. Based on this unit of interaction between the agent and environment, the agent sequentially performs further actions based on the states received from the environment.

In the formed environment, the agent's objective is to explore different sets of actions and exploit [10], [7], [24] the optimal actions maximizing the accumulated reward. To optimize the policy network of the agent, the RL algorithm is utilized with the rewards collected from the interaction with the environment; refer to Figure 3.

2.2. Bayesian Optimization for Hyperparameter Optimization

The approach, enhancing machine learning models through the optimal selection of hyperparameters of the **child network** by using Bayesian optimization [39], leverages a **probabilistic model** to predict the expected performance of child network on the validation dataset. As described in Figure 4, in the context of Bayesian Optimization, the probabilistic model is the deep learning model that decides a certain combination of hyperparameters (input of probabilistic model) applied to train the child network through the training dataset whose performance on the validation dataset (output of probabilistic model) is utilized for optimizing the probabilistic model. Through an iterative process that trains and evaluates a child network on the training and validation dataset with varying hyperparameter configurations decided by the probabilistic model, this method identifies the combination of hyperparameters best performing on the validation dataset. Bayesian optimization optimizes this process by using information from previous iterations to improve the search for effective hyperparameters, thereby balancing the exploration of new configurations against the exploitation of previously successful ones.

Commonly, both the suggested method leveraging the RL and the Bayesian optimization explore the set of hyperparameters that are utilized during the training of a child network and become optimized by their own algorithm to find the optimal hyperparameters that are expected to produce the highest performance on the validation dataset. Compared to the suggested method, the Bayesian optimization statically maintains the hyperparameters configured for the child network until the training of the child network with a certain set of hyperparameters is completed, as described in Diagram 1 in Figure 2, while the suggested method adaptively adjusts the hyperparameters according to the evolving state of the child network during the training process.

However, both the suggested method and Bayesian optimization are solely dependent on the performance evaluated by a validation dataset for optimization process, which can have the risk of decreased generalizability on the child network. This deterioration occurs when the model, trained with the hyperparameters selected by the Bayesian Optimization approach, can only perform well on a specific validation set, compromising its ability to generalize to unseen data. To counteract this, it is recommended to conduct a final evaluation on a separate test dataset.

3. Method

In this section, firstly, while utilizing the relationship between child network and agent, the overall description of the state, action and reward is presented. Secondly, the concept of $Chunk$ is presented to distinguish the propagations generated by the child network during the training or validation process. Thirdly, based on the concept $Chunk$, the notations, utilized for defining MDP, is presented. Finally, the MDP formulation is presented.

3.1. Overall description of MDP

Firstly, compared to the Bayesian Optimization explained in the related work, the **child network** in the suggested method, which should be separated from the policy network of **agent**, is any deep learning model trained with only training dataset under the regularization whose hyperparameters are decided by the agent; refer to Figure 3. The policy network of the agent is the Neural Network that produces a certain value for hyperparameters of regularization (output of policy network) based on the dynamically evolving state of the child network (input of policy network). Under the regularization whose hyperparameters are selected by the agent, the child network is trained, evaluated and tested with the training, validation and test dataset, respectively.

The state, action and reward are defined based on the relationship between the child network and the policy network of the agent. The state is defined as the array of the training loss generated during the training process of the child network. These sequences of losses can indirectly represent the status of the trained CNNs with high efficiency. The action is defined as the selection of regularization hyperparameters decided by the agent. After the child network is trained by regularization with certain hyperparameters during a certain period, the agent receives the reward, the child network's performance evaluated by F1 score on a validation dataset distinct from the training data.

3.2. Chunk

As described in Figure 5, $Chunk$ is a conceptual holder containing N steps of forward propagations generated by the child network during the training or validation process. This concept is utilized to specify the k-th forward propagation's order within the forward propagations contained in the t-th $Chunk$ where the k and t are denoted as the order of certain forward propagation among whole propagations and order of certain $Chunk$, respectively; refer to Figure 5. To mathematically express this, the relationship among N, k and t ($t \geq 1$) is defined in the Equation 2. By the relationship above, the k-th forward propagation corresponds to $k - (t-1) * N$ -th propagation in t-th $Chunk$.

To distinguish the forward propagations generated during the training the child network from propagations generated during validation, the $Chunk$ is subdivided further.

With the t-th $Chunk$ denoted as C_t, C_t^{train} is denoted as the $Chunk$ containing the propagations generated during training child network. In contrast, C_t^{val} is denoted as the $Chunk$ containing the propagations generated during validation process. The propagations generated during training and validation process are independently counted as described in Figure 6.

$$t = \begin{cases} \lfloor \frac{k}{N} \rfloor + 0 & \text{if } k \mod N = 0 \\ \lfloor \frac{k}{N} \rfloor + 1 & \text{if } k \mod N \neq 0 \end{cases} \tag{2}$$

3.3. Notation

The state, denoted as S_t, is defined by the array of training losses generated in a single chunk C_t^{train}. Training losses generated during C_t^{train} are collected and utilized to decide the hyperparameters determined before starting to proceed C_{t+1}^{train} and applied during C_{t+1}^{train}; refer to Figure 7. We denote $s_{t,i}\,(i = 1, 2, \dots, N)$ as a training loss of child network which is computed from the i-th forward propagation generated within the C_t^{train}.

The action, denoted as A_t, is to decide the hyperparameters before starting to proceed with the training process in C_{t+1}^{train}. These hyperparameters are applied during the training process in C_{t+1}^{train}. Respectively, we denote $[a, b]$, $a_{t,i}\,(i = 1, 2, \dots, h)$ and $\pi_\theta(S_{t+1}|S_t, A_t)$ as range where the hyperparameters can be configured, one of decided hyperparameters where the h is denoted as the number of adjustable hyperparameters and the probability distribution which the action A_{t+1} follows.

The next state, denoted as S_{t+1}, is also array of training losses generated under the selected regularization hyperparameters (A_t); refer to Figure 7.

The reward, denoted as R_t, is calculated by subtracting the mean of training F1 scores collected during C_{t+1}^{train} from the mean of validation F1 scores collected during C_{t+1}^{val}; refer to Figure 7. The m is the number of randomly sampled data from the validation dataset while the M is the total number of data in validation dataset. We denote $TF_{t,i}\,(i = 1, 2, \dots, N)$ as a F1 score evaluated on training dataset from the child network's i-th forward propagation inside the C_t^{train} and $VF_{t,i}\,(i = 1, 2, \dots, m)$ as a F1 score evaluated on validation dataset during C_t^{val}.

3.4. MDP formulation

The MDP can be described with a tuple: (S, A, R, γ), where 'S' stands for the set of states, 'A' for the set of actions, 'R' for the reward function, and 'γ' for the discount rate.

3.4.1 State

S_t is defined as the array of training losses generated by the child network during C_t^{train}. Although directly utilizing parameters of the model itself, as described in Diagram 3 inside Figure 2, is a straightforward approach for defining the state, it requires huge computation resources because the policy network, described in Figure 3, generally utilize the Deep Neural Network(DNN) to decide the suitable hyperparameters. Instead, because the individual training loss is indirectly correlated with the updated parameter of child network and sequentially generated training losses can reflect the trend of updated parameters, as described in Figure 8, the bundle of training losses can be used for representing the state of child network.

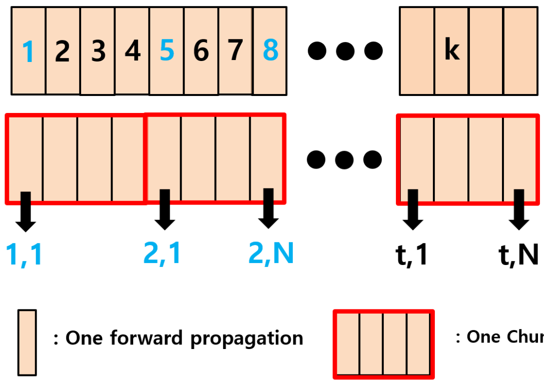

Figure 5. The upper Diagram shows the propagation generated along with the training process. The lower Diagram shows that every chunk in the training process consists of forward propagations(N=4).

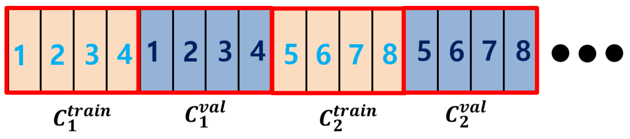

Figure 6. This figure shows the arrangement of C_t^{train} and C_t^{val} across the t.

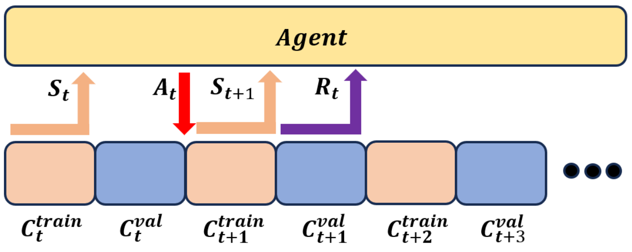

Figure 7. The image shows the interaction between the agent and the child network.

$$S_t = (s_{t,1}, s_{t,2}, \dots, s_{t,N}) \tag{3}$$

$$S_0 = (0, 0, \dots, 0)$$

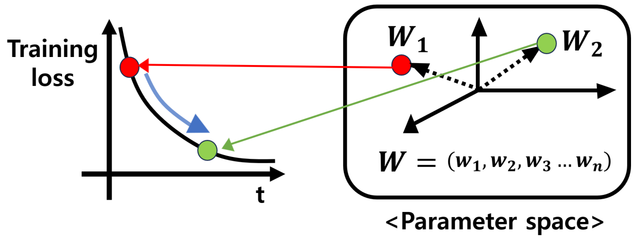

Figure 8. The image shows the decreasing training loss along with the updated parameters which correspond to the one of points in the parameter space

3.4.2 Action

The A_t undertaken by the agent is defined as the determination of specific values for regularization hyperparameters which also affect the next state, S_{t+1}. The regularization with those hyperparameters is applied during the training of the child network. The determined hyperparameters remain consistent throughout the C_{t+1}^{train}. This determination of hyperparameters is based on the status of the child network because the agent utilizes the child network's training losses collected during C_t^{train}. Also, how the hyperparameters are configured distinctively affect how the training losses are generated in C_{t+1}^{train}, which means S_{t+1} is dependent on the A_t given S_t; refer to Figure 3.

$$A_t = \{a_{t,1},\, a_{t,2},\, \dots,\, a_{t,h}\} \quad (4)$$
$$A_0 = \{a_{0,1}, a_{0,2}, \dots, a_{0,h}\}$$
$$where\ a_{0,i} \sim U(a,b)$$

$$h = |A_t| \quad (5)$$

$$A_{t+1} \sim \pi_\theta(S_{t+1}|S_t, A_t) \quad (6)$$

3.4.3 Reward

The R_t is defined by subtracting the mean of the training F1 score, collected during the C_{t+1}^{train}, from the mean validation F1 score collected from the C_{t+1}^{val}, as described by the Equation 7, 8, 9, 10 and 11. The N and m samples are randomly selected from training and validation datasets to calculate the means during C_{t+1}^{train} and C_{t+1}^{val}, respectively; refer to Figure 7.

By above definition, the policy network of agent can be optimized to select the regularization hyperparameters that can guide the child network both not to memorize the noise in the training dataset and to learn essential features. The objective of the agent is select the regularization hyperparameters that can maximize the reward. To increase the reward, the gap of performances between training dataset and validation dataset should not widen while performance of child network on the validation dataset increases. This is because while MTF_t^N tends to increase but has an upper limit, MVF_t^m, on the other hand, not only escalates continuously with bigger upper limit than the upper limit of MTF_t^N but also typically is greater than MTF_t^N. Therefore, in the situation where the child network is trained with only training dataset, the agent explores and exploits strategies that adjust the regularization hyperparameters which can ensure the child network does not memorize the noise in the training dataset while learning essential features.

$$TF_t^N = (TF_{t,1},\, TF_{t,2},\, \dots,\, TF_{t,N}) \quad (7)$$

$$VF_t^m = (VF_{t,1},\, VF_{t,2},\, \dots,\, VF_{t,m}) \quad (8)$$

$$MTF_t^N = \frac{\|TF_t^N\|_1}{N} \quad (9)$$

$$MVF_t^m = \frac{\|VF_t^m\|_1}{m} \quad (10)$$

$$R_t = MVF_{t+1}^m - MTF_{t+1}^N \quad (11)$$

4. Experiment

The following experiments show that the method proposed in this paper can distinctively prevent the deep learning model from memorizing the noise in the training dataset while facilitating the child network to learn the essential features. For the baseline, Dynamic Noise Injection (DNI) [13] which is already proven to be effective for In-the-wild FER domain is utilized. For the challengers against the baseline, the universally utilized regularizations such as noise injection and dropout are reinforced with the suggested method. We employ not only CIFAR100 [19] but also In-the-wild FER datasets such as FER2013 [16] and Affect Net [25]. In the case of In-the-wild FER, it is remarkable that enhancing model performance through regularization is challenging, as shown in the experiments from [13] and Table 1. For analysis, the graphical analysis, Grad-CAM analysis and Numerical analysis are presented.

4.1. Experimental Setup

4.1.1 Hardware and Software

All experiments were conducted using Pytorch. All the experiments were conducted using a GPU server equipped with two NVIDIA RTX 3090 GPUs, 128 GB RAM, and an Intel i9-10940X CPU.

	Vanilla	+NI	+D	+DNI(BL)	+RNI	+RD
CIFAR100,Resnet34	0.831	0.843	0.844	0.851	0.852	**0.875**
FER2013,Resnet34	0.615	0.602	0.601	0.618	0.616	**0.626**
AffectNet,Resnet34	0.489	0.480	0.486	0.506	0.515	**0.534**

Table 1. F1 scores on the test dataset across the dataset, model and regularization. NI=Noise Injection, D=Dropout, DNI=Dynamic Noise Injection [13], BL=Baseline, RNI=Reinforced Noise Injection, RD=Reinforced Dropout

4.1.2 Baseline and Challenger

For the baseline model, ResNet34 trained with the regularization termed Dynamic Noise Injection (DNI) suggested in [13] (also introduced in the introduction) is utilized. This regularization is especially developed for alleviating the overfitting arising in the domain of In-the-Wild FER.

For challengers against a baseline, ResNet34 trained with suggested regularization methods, which are named Reinforced Noise Injection (RNI) and Reinforced Dropout (RD), are utilized. RNI is the Noise Injection whose hyperparameter, the standard deviation used in the normal distribution to sample the noise, is adjusted by agent across the training process of child network. RD is the Dropout whose hyperparameter, the drop-rate specifying the fraction of neurons that are temporarily removed, is adjusted by agent across the training process of child network.

The training detail of the policy network of the agent is as follows: The agent network employs a Multi-Layer Perceptron (MLP) architecture and the Proximal Policy Optimization (PPO) algorithm is employed, with an epsilon value of 0.2, as outlined in [41]. The γ is configured as 0.995.

The training detail of the child network is as follows: The ResNet34 is utilized as the child network. The datasets used for training, validation and testing are CIFAR100, FER2013 and AffectNet. CIFAR100 is used for benchmarking image classification with 600 images in 100 categories. FER2013 features uncontrolled, grayscale images of faces with seven expressions, suited for real-world emotion recognition challenges. Affect Net contains images annotated with eight expressions and is also applicable in real-world scenarios. For experiments involving the CIFAR100 and FER 2013 datasets, the batch size was set to 64, and the learning rate was 0.001. In contrast, for the Affect Net dataset, a larger batch size of 256 and a lower learning rate of 0.0001 were used. For the loss function of the child network, this paper employs Cross-Entropy Loss with Adam optimizer. The primary metric for evaluating model performance was the F1 score. N and m are configured to both 20.

4.2. Result and Analysis

There are three types of analysis, graphical analysis, Grad-CAM analysis and the numerical analysis. As mentioned in the related work, the suggested method involves the optimization process of the policy network based on the child network's performance on the validation dataset. Consequently, there is a concern that rather, it might lead to an increase in generalization error due to potential bias towards a specific validation dataset. To check whether there is issue of this bias, the Grad-CAM and test datasets are utilized for evaluating the performance.

4.2.1 Graphical Analysis

The trends described below are always identical across the different dataset even though the graph in Figure 9 only shows the case of AffectNet.

As described in Figure 9, the training losses generated by the child network, trained under the regularization of RNI and RD, are always greater than the training losses generated by the baseline. Moreover, the validation losses generated by the child network, trained under the regularization of RNI and RD, are lower than or at least equal to validation losses generated by the baseline. These observations imply that the suggested method can effectively guide the trained deep learning model both not to memorize the noise in the training dataset and to learn essential features.

In addition, even though the training is continuously conducted, the training losses generated by the child network, trained under the regularization of RNI and RD, tends to decrease slowly while the validation losses tends not to increase. This observation is remarkable, given that the learning scheduler is not utilized at all. Through this observation, it can be also confirmed that the suggested method can be effective for preventing the trained deep learning model from memorizing the noise in the training dataset.

This effectiveness is possible because the policy network of agent, optimized by the RL algorithm, can **adaptively adjust the regularization hyperparameters expected to minimize overfitting according to the dynamically evolving state of the trained depp learning model**.

4.2.2 Grad-CAM Analysis

Given the issue of bias which can rather increase the generalization error by applying the suggested method, the Grad-CAM is utilized as one way to counter the aforementioned issue. Grad-CAM, short for Gradient-weighted Class Activation Mapping, is a technique for making the predictions of CNNs understandable to humans. It highlights the regions of an input image that are utilized for predictions in a specific class. The ordering of the child network's focus

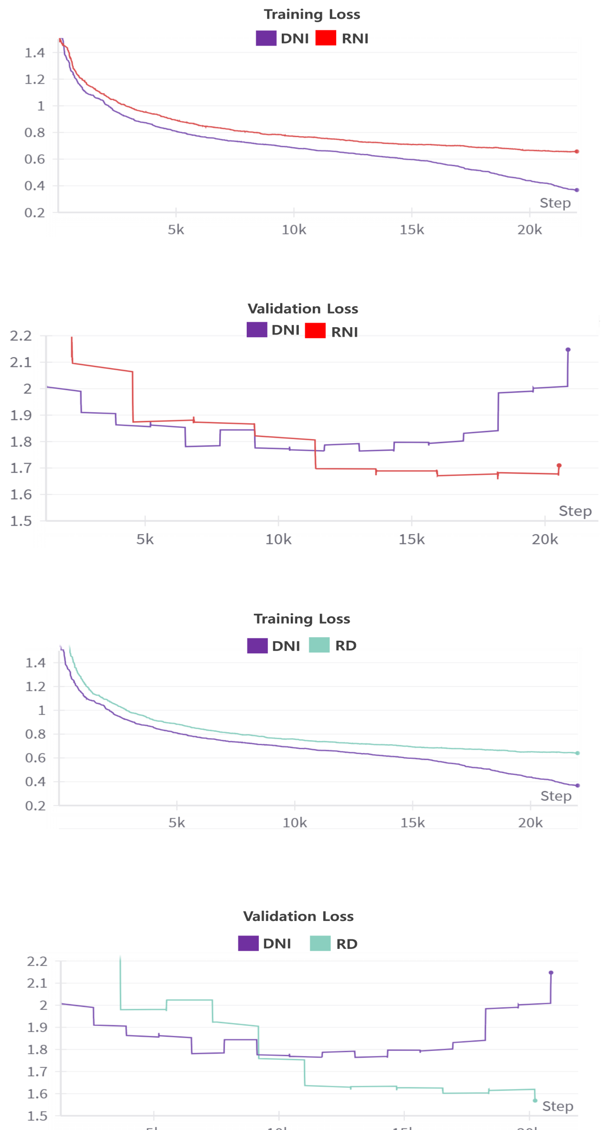

Figure 9. This figures show the training loss and validation loss graph on the AffectNet dataset.

areas—red, yellow, and blue—indicates that red represents the area where the model focuses the most, followed by yellow and blue.

Figure 10 illustrates that the DNI, unlike the RD, tends to memorize noise, as evidenced by its focus on the bubble around the face for predictions. In contrast, the RD does not concentrate on the bubble, indicating a relative lack of noise memorization. Moreover, while the RD utilizes the all parts of face, including the eyes, nose and mouth, others relatively does not utilize the mouth which is also crucial cue for recognizing the emotion. These observations

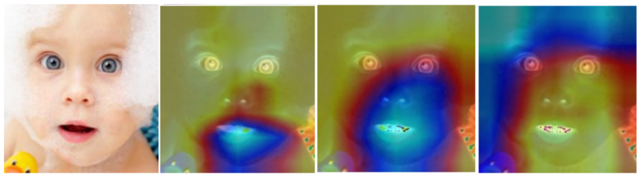

Figure 10. This figure shows that the result of Grad-CAM where the true class and the predicted class are identical in the training data. The far left image is the original image, and from the left in the second picture are the results of Grad-CAM for DNI, RNI, and RD.

strengthen the argument that the suggested method can effectively guide the trained deep learning model both not to memorize the noise in the training dataset and to learn essential features.

4.2.3 Numerical Analysis

Given the issue of bias which can rather increase the generalization error by applying the suggested method, the evaluation on the test dataset is utilized as another way to counter the aforementioned issue. The evaluations on the test dataset can be utilized to estimate the generalization capacity of the trained model. The performances on the test dataset are presented in the table 1. Especially, as described in the table, the RD consistently and distinctively outperforms other methods. This means the RD has lower generalization error than the baseline. This observation also can strengthens the argument that the suggested method can effectively guide the trained deep learning model both not to memorize the noise in the training dataset and to learn essential features.

5. Conclusion

This paper introduces a novel regularization method that leverages RL to adaptively adjust the regularization hyperparameters according to the dynamically changing state of CNNs during training. This approach is particularly designed to combat the challenge of overfitting in FER, especially pronounced in uncontrolled, in-the-wild environments where traditional methods falter due to the limited amount of data and noise inherent in such data. The proposed method's effectiveness is underscored by its ability to minimize the memorization of noise and learn essential features from facial expressions, thus significantly enhancing the CNNs' generalization. Experimental results on diverse datasets such as CIFAR100, FER2013, and AffectNet demonstrates the method's superiority over conventional regularization techniques, marking a significant step forward in the development of AI systems capable of accurately recognizing and interpreting human emotions from facial expressions.

References

[1] Baher Abdulhai and Lina Kattan. Reinforcement learning: Introduction to theory and potential for transport applications. *Canadian Journal of Civil Engineering*, 30(6):981–991, 2003. 2, 3

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee, 2017. 1

[3] Shun-ichi Amari. Backpropagation and stochastic gradient descent method. *Neurocomputing*, 5(4-5):185–196, 1993. 1

[4] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017. 2, 3

[5] Mohammad Mahdi Bejani and Mehdi Ghatee. A systematic review on overfitting control in shallow and deep neural networks. *Artificial Intelligence Review*, pages 1–48, 2021. 2

[6] Yoshua Bengio. Gradient-based optimization of hyperparameters. *Neural computation*, 12(8):1889–1900, 2000. 2

[7] Peter Dayan and Nathaniel D Daw. Decision theory, reinforcement learning, and the brain. *Cognitive, Affective, & Behavioral Neuroscience*, 8(4):429–453, 2008. 3

[8] Daniel Dewey. Reinforcement learning and the reward engineering principle. In *2014 AAAI Spring Symposium Series*, 2014. 3

[9] Jonas Eschmann. Reward function design in reinforcement learning. *Reinforcement Learning Algorithms: Analysis and Applications*, pages 25–33, 2021. 3

[10] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006. 3

[11] Benyamin Ghojogh and Mark Crowley. The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial. *arXiv preprint arXiv:1905.12787*, 2019. 1

[12] Bob Givan and Ron Parr. An introduction to markov decision processes. *Purdue University*, 2001. 2, 3

[13] SangHwa Hong and Jin-Woo Jeong. Dynamic noise injection for facial expression recognition in-the-wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5708–5714, 2023. 2, 6, 7

[14] Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. In *Compressed Sensing and Its Applications: Third International MATHEON Conference 2017*, pages 153–193. Springer, 2019. 1

[15] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285, 1996. 2

[16] Yousif Khaireddin and Zhuofa Chen. Facial emotion recognition: State of the art performance on fer2013. *arXiv preprint arXiv:2105.03588*, 2021. 6

[17] W Bradley Knox and Peter Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *AAMAS*, pages 5–12, 2010. 3

[18] W Bradley Knox and Peter Stone. Reinforcement learning from simultaneous human and mdp reward. In *AAMAS*, pages 475–482. Valencia, 2012. 3

[19] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[20] Jan Kukačka, Vladimir Golkov, and Daniel Cremers. Regularization for deep learning: A taxonomy. *arXiv preprint arXiv:1710.10686*, 2017. 1

[21] Jake Lever, Martin Krzywinski, and Naomi Altman. Points of significance: model selection and overfitting. *Nature methods*, 13(9):703–705, 2016. 1

[22] Shan Li and Weihong Deng. Deep facial expression recognition: A survey. *IEEE transactions on affective computing*, 13(3):1195–1215, 2020. 1

[23] Renjie Liao, Yuwen Xiong, Ethan Fetaya, Lisa Zhang, Ki-Jung Yoon, Xaq Pitkow, Raquel Urtasun, and Richard Zemel. Reviving and improving recurrent back-propagation. In *International Conference on Machine Learning*, pages 3082–3091. PMLR, 2018. 2

[24] Pattie Maes, Maja J Mataric, Jean-Arcady Meyer, Jordan Pollack, and Stewart W Wilson. Explore/exploit strategies in autonomy. 1996. 3

[25] Ali Mollahosseini, Behzad Hasani, and Mohammad H. Mahoor. Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 10(1):18–31, 2019. 6

[26] Attila Nagy and Ábel Boros. Improving the sample-efficiency of neural architecture search with reinforcement learning. *arXiv preprint arXiv:2110.06751*, 2021. 3

[27] Hyeonwoo Noh, Tackgeun You, Jonghwan Mun, and Bohyung Han. Regularizing deep neural networks by noise: Its interpretation and optimization. *Advances in neural information processing systems*, 30, 2017. 1

[28] Martin L Puterman. Markov decision processes. *Handbooks in operations research and management science*, 2:331–434, 1990. 2, 3

[29] Steffen Rendle. Learning recommender systems with adaptive regularization. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 133–142, 2012. 2

[30] Rudy Semola, Julio Hurtado, Vincenzo Lomonaco, and Davide Bacciu. Adaptive hyperparameter optimization for continual learning scenarios. *arXiv preprint arXiv:2403.07015*, 2024. 1

[31] Olivier Sigaud and Olivier Buffet. *Markov decision processes in artificial intelligence*. John Wiley & Sons, 2013. 2, 3

[32] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999. 2

[33] Yingjie Tian and Yuqi Zhang. A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, 80:146–166, 2022. 1

[34] Gal Vardi. On the implicit bias in deep-learning algorithms. *Communications of the ACM*, 66(6):86–93, 2023. 3

[35] Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. Regularization of neural networks using dropconnect. In *International conference on machine learning*, pages 1058–1066. PMLR, 2013. 1

[36] Jiazhuo Wang, Jason Xu, and Xuejun Wang. Combination of hyperband and bayesian optimization for hyperparameter optimization in deep learning. *arXiv preprint arXiv:1801.01596*, 2018. 1

[37] Douglas J White. A survey of applications of markov decision processes. *Journal of the operational research society*, 44(11):1073–1096, 1993. 2, 3

[38] Marco A Wiering and Martijn Van Otterlo. Reinforcement learning. *Adaptation, learning, and optimization*, 12(3):729, 2012. 2

[39] Jia Wu, Xiu-Yun Chen, Hao Zhang, Li-Dong Xiong, Hang Lei, and Si-Hao Deng. Hyperparameter optimization for machine learning models based on bayesian optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019. 4

[40] Xue Ying. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, page 022022. IOP Publishing, 2019. 1

[41] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022. 7

[42] Tong Yu and Hong Zhu. Hyper-parameter optimization: A review of algorithms and applications. *arXiv preprint arXiv:2003.05689*, 2020. 2

[43] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3):107–115, 2021. 1

[44] Han Zhao, Yao-Hung Hubert Tsai, Russ R Salakhutdinov, and Geoffrey J Gordon. Learning neural networks with adaptive regularization. *Advances in Neural Information Processing Systems*, 32, 2019. 2