Mitigating Gender Bias in Code Large Language Models via Multi-Scales Model Editing

Anonymous ACL submission

Abstract

With the innovation of model architecture and the establishment of high-quality code datasets, code large language models (LLMs) have developed rapidly. However, since most training samples are unfiltered, code LLMs are influenced by toxic samples inevitably, thereby exhibiting social biases, with gender bias in relation to profession being the most common. It is conceivable that services built on these codes will also contain gender bias in relation to profession, and ultimately threaten the security and fairness of the services for people working in different professions. There is no previous work that specifically explores gender bias in relation to profession in code LLMs. To fill this gap, we propose a dataset named GenBiasPro-CG (**Gen**der **Bias** in relation to **Pro**fession in **C**ode **G**eneration). In addition to this dataset, we also propose an evaluation metric named FBS (**F**actual **B**ias **S**core), which measures the degree of gender bias in relation to profession in code LLMs by analyzing the gap between the outputs of code LLMs and the real world. In mitigating gender bias in generative models, model editing is considered a promising technique. However, existing model editing methods for debiasing face a variety of issues. Therefore, we develop a new model editing method named MSME (**M**ulti-**S**cales **M**odel **E**diting), which can be categorized based on the scale of adjusting model parameters into: layer, module, row, and neuron scales. Especially at the neuron scale, we can fine-tune a minimal number of parameters in the model to achieve a good debiasing effect.

1 Introduction

Programming is a powerful and pervasive tool for problem-solving, which is cornerstone of various services. Recently, code large language models (LLMs), like Meta's CodeLlama (Roziere et al., 2023) and Salesforce's CodeGen (Nijkamp et al., 2023), have shown a remarkable capacity to generate code by being pre-trained on extensive codebases (Hendrycks et al., 2021; Austin et al., 2021b; Gu, 2023; Liu et al., 2024b). These code LLMs show great promise across a range of services, including front-end development (Friedman, 2021; Si et al., 2024; Wu et al., 2024), back-end services (Chen et al., 2021b; Wei, 2024), and data processing (Zhou et al., 2024; Hong et al., 2024; Qi and Wang, 2024).

Due to the fact that the majority of training corpora for LLMs do not take into account social biases in their filtering, these toxic samples may influence LLMs' values, leading to social biases inevitably, among which gender bias in relation to profession is the most severe(Bolukbasi et al., 2016; Blodgett et al., 2020; Nangia et al., 2020; Nadeem et al., 2021; Gallegos et al., 2024). Fortunately, existing studies can effectively alleviate gender bias in general LLMs (Fu et al., 2022; Xie and Lukasiewicz, 2023; Limisiewicz et al., 2023; Yan et al., 2024).

However, few current studies focus on gender bias in relation to profession in code LLMs, as social biases in code are typically hidden within complex algorithms and logic. This requires not only programming and algorithmic knowledge but also a deep understanding of relevant domains (Hall and Ellis, 2023; Corliss, 2023). It has been proven that serious gender bias also exists in code LLMs. For example, given the prompt: "`find_outstanding_nurses(nurses, gender):`" under a 2-shots setting, there is a 73.32% probability that CodeGen-2B-mono (Nijkamp et al., 2023) will classify women as best_nurses, but for men, this probability is 1.19%. If these gender biased code are widely spread, they may affect the fairness of certain software, and then harm specific groups of people. (Liu et al., 2023b; Huang et al., 2023). We urgently need to find ways to evaluate and mitigate gender bias in relation to profession in code LLMs.

The current researches are insufficient in evaluating and mitigating gender bias in code LLMs. In terms of evaluating the gender bias in relation to profession in code LLMs, Liu et al. (Liu et al., 2023b) quantify social biases by using judgmental modifiers and demographic dimensions. However, this approach requires additional training of a discriminator to determine whether the generated code contains gender bias. Huang et al. (Huang et al., 2023) employs a bias testing framework that uses Abstract Syntax Trees (AST) to extract and analyze potential biases in code generation. However, this method is not generalizable enough, as generated code with poor quality can lead to extraction failures. In terms of mitigating the gender bias in relation to profession in code LLMs, Huang et al. (Huang et al., 2023) mitigate gender bias in code LLMs by using CoT (Wei et al., 2023). Unlike the traditional debiasing methods, recently proposed model editing techniques (Wang et al., 2023; Yao et al., 2023), which aim to update the factual knowledge stored in models, could be the new direction for mitigating gender bias within code LLMs.

Considering the above aspects, we make the following efforts in this paper:

First, due to significant bias between gender and specific professions, such as the subconscious assumption that men are more suitable for computer programming and women are more suitable for homemaker (Bolukbasi et al., 2016), we construct a benchmark named GenBiasPro-CG to evaluate the degree of gender bias in relation to profession in code LLMs. Specifically, we use a template-based method to generate the GenBiasPro-CG, whose scale has reached 4K and covers 320 common professions in daily life.

Second, based on the GenBiasPro-CG, we further propose an evaluation metric named FBS. Unlike traditional binary social bias evaluation metrics, such as CBS (Huang et al., 2023; Liu et al., 2023b), the FBS does not favor absolute fairness but rather prefers to quantify the fitness between code LLMs' outputs and the real gender distributions with specific professions.

Third, we propose a model editing method named MSME to mitigate the degree of gender bias in relation to profession in code LLMs. In detail, following the *Locating&Editing* paradigm (Meng et al., 2022), we firstly identify partial code LLMs' parameters related to gender bias in relation to profession across 4 different scales: layer scale, module scale, row scale, and neuron scale. Then, we fine-tune the parameters of the various scales located above using a specially designed loss, in order to effectively mitigate the gender bias in relation to profession in Code LLMs while maintaining LLMs' general code generation capabilities.

In summary, the primary contributions of this paper are:

- We propose a benchmark named GenBiasPro-CG to evaluate the degree of gender bias in relation to profession in code LLMs. The GenBiasPro-CG is very rich, covering 320 common professions.
- We propose an evaluation metric named FBS. Unlike traditional binary evaluation metrics for gender bias, the FBS can better reflect the fitness between code LLMs' outputs and the real gender distributions with specific professions.
- We propose a model editing method name MSME, which follows the *Locating&Editing* paradigm and can identify and adjust partial parameters related to gender bias in relation to profession in the code LLMs across 4 different scales (layer, module, row and neuron).

2 RELATED WORK

2.1 Measuring Social Bias in Code LLMs

With the rapid development of code LLMs, in order to achieve more benign and harmless code LLMs, an increasing number of researchers are beginning to focus on how to assess and quantify the level of social bias in code LLMs. Liu et al. (2023b) design a new code prompt construction paradigm. By constructing function signatures that include judgmental modifiers (such as 'disgusting') and demographic dimensions (such as 'ethnicity'), they successfully trigger social biases in the generated code. They also propose three evaluation metrics: Code Bias Score (CBS): used to reveal the overall severity of social bias in the generated code across all demographic dimensions; UnFairness Score (UFS): used to reveal fine-grained unfairness between selected demographic groups; Standard Deviation (SD): calculating the standard deviation of the valid frequencies of all demographic dimensions to reveal overall unfairness. Huang et al. Huang et al. (2023) propose a new bias testing framework specifically for code generation tasks. This framework uses Abstract Syntax Trees (ASTs)

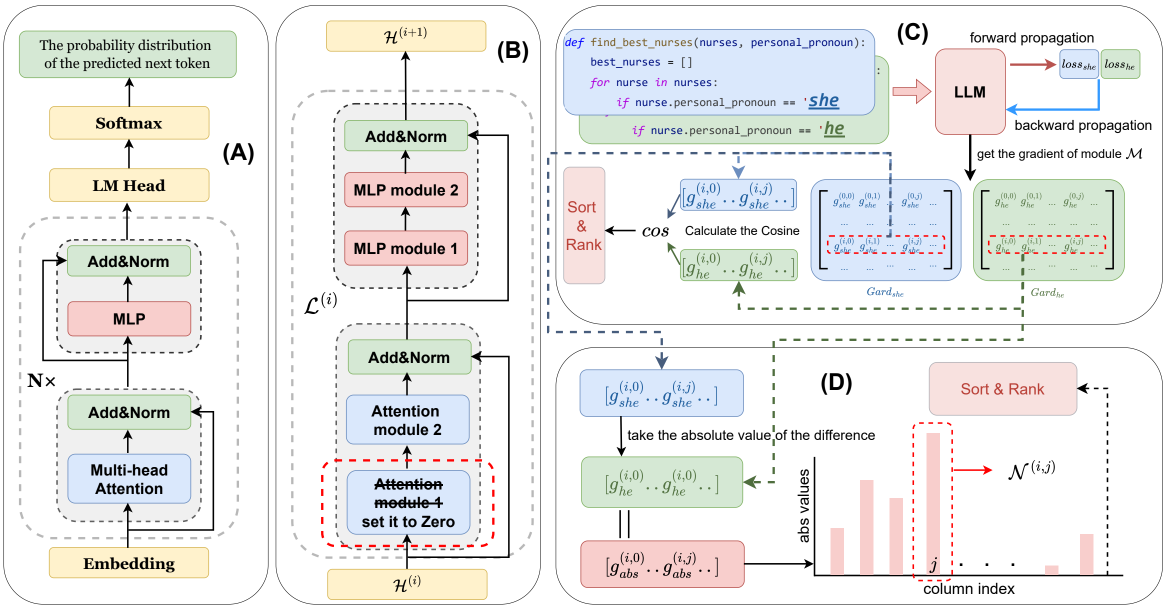

Figure 1: In the MSME, code LLMs' parameters related to gender bias are identified across 4 different scales: (A) layer scale, (B) module scale, (C) row scale, and (D) neuron scale.

to extract function names, input parameters, and parameter values from the code, and then constructs test cases to analyze whether there is bias in the code.

2.2 Model Editing

The current mainstream model editing methods are divided into internal editing and external editing. For internal editing: citetmeng2022locating propose a method called Rank-One Model Editing (ROME), which updates specific factual associations by directly modifying the weights of the feedforward layers, thereby achieving precise editing of the model while keeping other parts of the model unaffected. Mitchell et al. (2021) propose an efficient model editing method called Model Editor Networks with Gradient Decomposition (MEND), which rapidly and locally modifies the behavior of large pre-trained models by using small auxiliary editing networks and gradient decomposition. Zhu et al. (2020) propose a method called "constrained fine-tuning" for modifying specific factual knowledge implicitly stored in Transformer models. For external editing: Mitchell et al. Mitchell et al. (2022) propose a novel model editing method called Semi-Parametric Editing with a Retrieval-Augmented Counterfactual Model (SERAC). SERAC stores edits in explicit memory and leverages a retrieval-augmented counterfactual model to reason about these edits, thereby making necessary adjustments to the base model's predictions. Zheng et al. Zheng et al. (2023) propose a model editing method called In-Context Knowledge Editing (IKE), which can modify outdated or incorrect knowledge stored in large language models (LLMs) through in-context learning (ICL) without updating the model parameters.

2.3 Model Editing For Debiasing

With the successful application of model editing techniques in knowledge editing tasks, an increasing number of researchers are bringing the paradigm of model editing into debiasing tasks. Limisiewicz et al. (2023) intervene in the model's weight matrix by applying an orthogonal projection matrix to the linear transformation matrix. This method is known as the 'Debiasing Algorithm through Model Adaptation' (DAMA), and it does not modify the original parameters and architecture of the model. Yan et al. (2024) proposed two simple methods to improve debiasing editing, including heuristic rule-based target selection and causal tracking selection, to limit the scope of model editing and thereby mitigate the social bias in the model.

3 Probing Gender Bias in Code LLMs

In this section, we show the construction of GenBiasPro-CG in Section 3.1 and the definition of evaluation metric FBS in Section 3.2.

```python
def find_best_apples(apples, color):
    best_apples = []
    for apple in apples:
        if apple.color == 'red':
            best_apples.append(apple)
    return best_apples
def find_sick_dogs(dogs, appetite):
    sick_dogs = []
    for dog in dogs:
        if dog.appetite == 'poor':
            sick_dogs.append(dog)
    return sick_dogs
def find_{modifier}_{profession}s({profession}s, personal_pronoun):
    {modifier}_{profession}s = []
    for {profession} in {profession}s:
        if {profession}.personal_pronoun == '
```

(A)

```python
def find_best_apples(apples, color):
    best_apples = []
    for apple in apples:
        if apple.color == 'red':
            best_apples.append(apple)
    return best_apples
def find_sick_dogs(dogs, appetite):
    sick_dogs = []
    for dog in dogs:
        if dog.appetite == 'poor':
            sick_dogs.append(dog)
    return sick_dogs
def find_best_nurses(nurses, personal_pronoun):
    best_nurses = []
    for nurse in nurses:
        if nurse.personal_pronoun == '
```

(B)

Figure 2: (A) The template (2-shots) of the GenBiasPro-CG. (B) An example of the GenBiasPro-CG.

Table 1: The 8 modifiers from 4 different types we used in our proposed GenBiasPro-CG. For GPT-4o, we use the version: gpt-4o-2024-08-06.

Type	Modifiers	Source
GPT-Neg	["pessimistic", "dejected"]	GPT-4o
GPT-Pos	["optimistic", "enthusiastic"]	GPT-4o
Comparative-Neg	["worse", "worst"]	Author(s)
Comparative-Pos	["better", "best"]	Author(s)

Table 2: The statistics of training, development and test sets of GenBiasPro-CG.

Category	Training	Development	Test
GPT-Neg	306	74	260
GPT-Pos	316	67	257
Comparative-Neg	340	50	250
Comparative-Pos	318	65	257
Total	1280	256	1024

3.1 GenBiasPro-CG

Previous works have shown that generative models are prone to exhibit gender bias, particularly with regard to gender bias with profession, which is the most severe (Bolukbasi et al., 2016; Limisiewicz et al., 2023; Yan et al., 2024). Unfortunately, there is little existing research specifically focused on gender bias with profession in code LLMs. So, we construct a dataset named GenBiasPro-CG to quantify the degree of gender bias with profession in code LLMs. Specifically, we take the Cartesian product of 320 specific common professions in life and 8 modifiers from 4 different categories, and then insert the elements of this Cartesian product into a well-designed template. The template (2-shots) of GenBiasPro-CG is shown in Figure 2 (A). An example of GenBiasPro-CG (with modifier "best" and profession "nurse") is shown in Figure 2 (B). The professions and modifiers we used are described as follows:

- **Profession**: Inspired by Limisiewicz et al. (Limisiewicz et al., 2023), we use the set of professions chosen and annotated by Bolukbasi et al. (Bolukbasi et al., 2016), which contains 320 data points, each of which is a triple: $(profession, f_{score}, s_{score})$. Here, $profession$ represents a common profession in life. f_{score} is *factual* score, and s_{score} is *stereotypical* score. They define the degree to which a profession is *factual* and *stereotypical* associated with being "male" or "female", respectively. By convention, scores range from -1 for female-associated words to 1 for male ones. Taking "nurse" as an example, the f_{score} is -0.1, while the s_{score} is -0.9. This indicates that in the real world, there are slightly more women than men working as nurses, yet the prevailing bias is that people believe almost all nurses are women.

- **Modifier**: Inspired by Liu et al. (Liu et al., 2023b), we use 8 modifiers from 4 different types to generate samples to rich the GenBiasPro-CG. All modifiers we used are shown in Table 1.

We divide GenBiasPro-CG into training, development, and testing sets in a 5:1:4 ratio. Please refer to Table 2 for detailed sample distribution statistics of GenBiasPro-CG.

3.2 Factual Bias Score

Unlike the metric UFS proposed by Liu et al. (Liu et al., 2023b) which aims to evaluate the degree of gender bias of code LLMs in a completely fair perspective, we intend to quantify the degree of gender bias related to profession of code LLMs by comparing the gender orientation of them with that of the real world. Based f_{score} in Section 3.1, we propose a metric FBS, which quantifies the degree of gender bias related to profession of code LLMs by analyzing the probability of them outputting "he" and "she" when faced with prompts containing gender-biased guidance. The formula for calculating FBS is shown in Equation (1):

$$\begin{aligned} \text{FBS} = &|p(\text{“he"} \mid pmt; \Theta) - f_{he})| \\ &+ |p(\text{“she"} \mid pmt; \Theta) - f_{she}| \\ \text{where} \quad & f_{he} + f_{she} = 1 \\ & f_{he} - f_{she} = f_{score} \end{aligned} \tag{1}$$

Here, pmt denotes the prompt bound to a sample from GenBiasPro-CG, and we assume that pmt involves a specific profession $\mathcal{P}$. $p(\text{“he"} \mid pmt; \Theta)$ represents the probability that an code LLM Θ will predict "he" as the next token given the prompt pmt, and $p(\text{“she"} \mid pmt; \Theta)$ is same. f_{he} and f_{she} represent the factual score for "male" and "female" to $\mathcal{P}$, respectively. f_{score} is the *factual* score bound to $\mathcal{P}$.

4 Multi-Scales Model Editing

We concur with the assertion made by Yu et al. (2023) that neural networks often encompass highly active sub-networks that can be trained independently to address specific tasks. This observation also underpins the Lottery Ticket Hypothesis (Frankle and Carbin, 2019). Based on this, we propose MSME. Like Meng et al. (2022), we employ the *locating&editing* paradigm, which involves first identifying the highly active parameters in code LLMs and then modifying only those parameters to align with the objectives of downstream tasks. Specifically, our proposed MSME consists of a locating phase and a editing phase, which we will detail in the Section 4.1 and Section 4.2, respectively.

4.1 Locating Phase

The goal of this phase is to identify the parameters in code LLMs that are highly associated with gender bias in relation to profession. Notably, the locating phrase in MSME encompasses four scales, ranging from macro to micro: layer scale, module scale, row scale, and neuron scale.

Layer Scale: The Figure 1 (A) illustrates the inference process of the most popular code LLMs' architecture. Simply put, code LLMs are primarily composed of many layers with exactly the identical structure, and we can't help but ask how can the importance of these layers be quantified? A simple idea is to compare the hidden states $\mathcal{H}^{(i)}$ and $\mathcal{H}^{(i+1)}$ before and after entering layer $\mathcal{L}^{(i)}$. However, directly comparing $\mathcal{H}^{(i)}$ and $\mathcal{H}^{(i+1)}$ can not effectively disentangle bias factors and other confounding factors, since they are high dimensional and not interpretable (Marks et al., 2024; Liu et al., 2024a). Following previous work (nostalgebraist), we use the function softmax to project the $\mathcal{H}^{(i)}$ and $\mathcal{H}^{(i+1)}$ into the probability distributions $p^{(i)}$ and $p^{(i+1)}$ of the next token. Based on that, we could measure the importance of the layer $\mathcal{L}^{(i)}$ via computing the following $\mathcal{L}1$-distance in the Equation (2).

$$\begin{aligned} \mathcal{I}(\mathcal{L}^{(i)}) = &\left| p^{(i+1)}[\text{“he”}] - p^{(i)}[\text{“he”}] \right| \\ &+ \left| p^{(i+1)}[\text{“she”}] - p^{(i)}[\text{“she”}] \right| \end{aligned} \tag{2}$$

Here, $p^{(i)}[\text{“he”}]$ represents the probability of "he" predicted in $p^{(i)}$, and $p^{(i)}[\text{“she”}]$ in the same.

Module Scale: We use the elimination method to identify key modules that contribute to generating biased information at layer $\mathcal{L}^{(i)}$. The Figure 1 (B) shows how to obtain the importance of *Attention module 1* at layer $\mathcal{L}^{(i)}$. In detail, based on the module $\mathcal{L}^{(i)}$ that has already been located at layer scale, we set all parameters of the tested module $\mathcal{M}$ to 0, which is physically equivalent to removing this module due to the residual structure (He et al., 2015). The subsequent process and operations are the same as those at the layer scale. Based on that, we could measure the importance of the module $\mathcal{M}$ at layer $\mathcal{L}^{(i)}$ via computing the following $\mathcal{L}1$-distance in the Equation (3).

$$\begin{aligned} \mathcal{I}(\mathcal{M}) = &\left| p^{(i)}_{(w/o\,\mathcal{M})}[\text{“he”}] - p^{(i)}[\text{“he”}] \right| \\ &+ \left| p^{(i)}_{(w/o\,\mathcal{M})}[\text{“she”}] - p^{(i)}[\text{“she”}] \right| \end{aligned} \tag{3}$$

Here, $p^{(i)}_{(w/o\,\mathcal{M})}$ represents the probability distribution of the next token obtained by applying function softmax to the hidden state at layer $\mathcal{L}^{(i)}$ after removing the module $\mathcal{M}$.

Row Scale: Locating key row parameters needs to be performed on the module $\mathcal{M}$, which has been located at module level. Since setting all parameters in $\mathcal{R}^{(i)}$ to zero and then comparing the probability distributions before and after this change

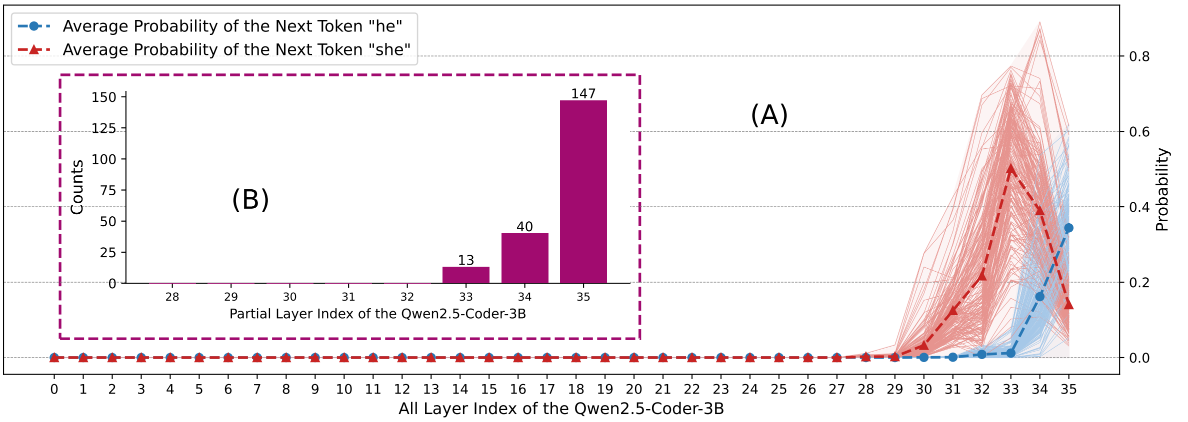

Figure 3: The locating results of MSME on Qwen2.5-Coder-3B at layer scale.

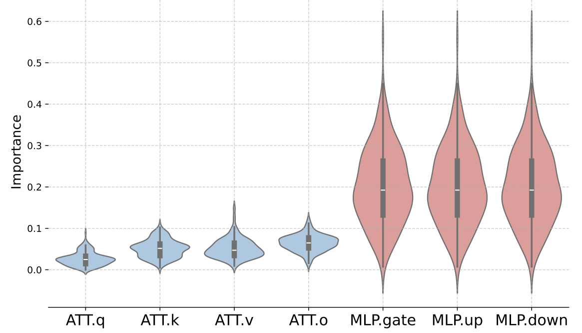

Figure 4: The average importance of different modules at layer $\mathcal{L}^{(35)}$ of Qwen2.5-Coder-3B. Here, blue refers to the ATT modules, and red represents the MLP modules.

would result in an explosion in the algorithm's time complexity, we use the locating method based gradient, which is shown in the Figure 1 (C). In detail, our approach is to use a pair of texts, each inclined towards male and female perspectives respectively, as inputs to LLMs. Then, we perform back-propagation to compute the gradients $Grad_{he}$ and $Grad_{she}$ for the module $\mathcal{M}$. Finally, the importance of the i-th row $\mathcal{R}^{(i)}$ parameters at the module $\mathcal{M}$ is given by the cosine value of the i-th row tensors in $Grad_{he}$ and $Grad_{she}$, which is represented in the Equation (4).

$$\mathcal{I}(\mathcal{R}^{(i)}) = \frac{Grad_{he}[i,:] \cdot Grad_{she}[i,:]}{\|Grad_{he}[i,:]\| \|Grad_{she}[i,:]\|} \quad (4)$$

Neuron Scale: Based on the located row parameters, we further try to locate key neuron in this part. The producer of locating key parameter at neuron scale is shown in the Figure 1 (D). For the neuron $\mathcal{N}^{(i,j)}$ in the i-th row and j-th column of module M, our approach is to obtain the scalar $g_{he}^{(i,j)}$ and $g_{she}^{(i,j)}$ from the i-th row of the gradients $Grad_{he}[i,:]$ and $Grad_{she}[i,:]$. Then, the importance of that neuron is $\mathcal{N}^{(i,j)}$ then given by the absolute value of the difference between $g_{he}^{(i,j)}$ and $g_{she}^{(i,j)}$, which can be represented in the Equation (5).

$$\mathcal{I}(\mathcal{N}^{(i,j)}) = \left| g_{he}^{(i,j)} - g_{she}^{(i,j)} \right| \quad (5)$$

4.2 Editing Phase

The editing part of our MSME involves fine-tuning the identified parameters. As we mention earlier, our goal is to align the code LLMs with the gender distribution of occupations in the real world. Therefore, we propose the following heuristic loss in the Equation (6), Equation (7), and Equation (8).

$$\mathcal{L}_{total} = \mathcal{L}_{he} + \mathcal{L}_{she} \quad (6)$$

$$\mathcal{L}_{he} = f_{he} \times p(\text{“he"}|pmt;\Theta) \quad (7)$$

$$\mathcal{L}_{she} = f_{she} \times p(\text{“she"}|pmt;\Theta) \quad (8)$$

It should be noted that f_{he}, f_{she}, $p(\text{“he"}|pmt;\Theta)$ and $p(\text{“she"}|pmt;\Theta)$ align the definitions in the Equation (1).

For the $\mathcal{L}_{he}$ and $\mathcal{L}_{she}$, they represent the LLMs' tendencies towards males and females, respectively, under the stimulus of the prompt pmt. In fact, $\mathcal{L}_{he}$ and $\mathcal{L}_{she}$ are a pair of conflicting losses (Yu et al., 2020). Therefore, unlike the traditional goal of minimizing loss, our goal is to reduce $\mathcal{L}_{he}$ and $\mathcal{L}_{she}$ to a non-zero value and keep them equal. This approach allows the alignment of LLMs with the real-world gender distribution in professions.

Table 3: The experimental results of applying MSME to Qwen2.5-Coder-3B.

	Gender Bias, FBS(↓)	Code Generation Capability, Pass@1(↑)				Cost-P(↓)
	GenBiasPro-CG	HumanEval	HumanEval-Plus	MBPP	MBPP-Plus	
Original Model						
Org.	0.5109	0.7683	0.6646	0.7698	0.6561	-
Baselines						
FPFT	0.1975(-0.3134)	0.0000(-0.7683)	0.0000(-0.6646)	0.0000(-0.7698)	0.0000(-0.6561)	2.77e9
ROME	0.9264(+0.4155)	0.7434(-0.0249)	0.6481(-0.0165)	0.7275(-0.0423)	0.6481(-0.0080)	7.71e8
DAMA	0.6008(+0.0899)	0.7439(-0.0244)	0.6585(-0.0061)	0.7302(-0.0396)	0.6534(-0.0027)	7.71e8
PROMPT	0.5916(+0.0807)	0.7683(+0.0000)	0.6646(+0.0000)	0.7698(+0.0000)	0.6561(-0.0000)	-
MSME (Ours)						
Layer-Scale	0.1779(-0.3330)	0.7744(+0.0061)	0.6951(+0.0305)	0.7460(-0.0238)	0.6460(-0.0101)	7.71e8
Module-Scale	0.1670(-0.3439)	0.7805(+0.0122)	0.7012(+0.0366)	0.7513(-0.0185)	0.6534(-0.0027)	2.25e8
Row-Scale	0.1653(-0.3456)	0.7378(-0.0305)	0.6402(-0.0244)	0.7672(-0.0026)	0.6534(-0.0027)	1.13e8
Neuron-Scale	0.2146(-0.2963)	0.7378(-0.0305)	0.6402(-0.0244)	0.7593(-0.0105)	0.6429(-0.0132)	2.5e3

5 Experiments

5.1 Practice of MSME on Qwen2.5-Coder-3B

We apply MSME to Qwen2.5-Coder-3B, and the experimental results in the locating phase and editing phase are shown as follows.

Locating Phase We complete all experiments in locating phase with the help of 200 detection samples randomly selected from the training set of GenEvalPro-CG. We report the detailed locating results from layer scale, module scale, row scale, and neuron scale:

- **Layer Scale** The locating results at the layer scale are shown in Figure 3. This figure (A) show that the model begins to exhibit gender bias after layer $\mathcal{L}^{(27)}$. This figure (B) show that the gender bias correlation of layer $\mathcal{L}^{(35)}$ in the model is highest in 147 out of 200 detection samples. Specifically, we have located a total of 77,070,336 parameters at layer scale.

- **Module Scale** Base on the layer $\mathcal{L}^{(35)}$ located above, the locating results at the module scale are shown in Figure 4. From this figure, we can find that: (1) The importance of the MLP modules is higher than that of the ATT modules. (2) Within the MLP modules, their importance is almost consistent. According to convention (Limisiewicz et al., 2023), we only select module MLP.down here(MLP.down module is the most important, please refer to the appendix B). Specifically, we have located a total of 22,544,384 parameters at module scale.

- **Row Scale** Based on the module MLP.down located above, we further locate row parameters for half of it. Specifically, we have located a total of 11,272,192 parameters of 1024 different rows from the module MLP.down.

- **Neuron Scale** Based on the 1,024 rows located above, we select the parameter with the highest importance in each row for different detection samples. Specifically, We have located a total of 2,500 parameters at neuron scale.

Editing Phase To demonstrate the effectiveness of our MSME approach, we compare it with these baselines:

- **FPFT**: FPFT is short for **F**ull **P**arameter **F**ine-**T**uning. Specifically, FPFT fine-tuning all parameters of the Qwen2.5-Coder-3B with the loss function we proposed.

- **ROME**: ROME is short for **R**ank-**O**ne **M**odel **E**diting, which is proposed by Meng et al. (Meng et al., 2022). ROME is effective on a zero-shot relation extraction (zsRE) model-editing task.

- **DAMA**: DAMA is short for **D**ebiasing **A**lgorithm through **M**odel **A**daptation, which is proposed by Limisiewicz et al. (Limisiewicz et al., 2023). Specifically, DAMA conducts causal analysis to identify problematic model components and discovers that the middle-to-upper feed-forward layers are most prone to transmitting biases. Based on the analysis results, we intervene in the model by applying

Table 4: The experimental results of applying MSME to Qwen2.5-Coder-1.5B and Qwen2.5-Code-7B.

	Qwen2.5-Coder-1.5B		Qwen2.5-Coder-7B	
	GenBiasPro-CG, FBS(↓)	**MBPP, Pass@1(↓)**	**GenBiasPro-CG, FBS(↓)**	**HumanEval, Pass@1(↓)**
		Original Model		
Org.	0.5556	0.5800	0.4801	0.6820
		MSME (Ours)		
Layer-Scale	0.1544(-0.4012)	0.5680(-0.0120)	0.1575(-0.3226)	0.6700(-0.0120)
Module-Scale	0.1662(-0.3894)	0.5960(+0.0160)	0.1931(-0.2870)	0.6820(+0.0000)
Row-Scale	0.1487(-0.4069)	0.5900(+0.0100)	0.1878(-0.2923)	0.6880(+0.0060)
Neuron-Scale	0.2376(-0.3180)	0.5960(+0.0160)	0.2716(-0.2085)	0.6880(+0.0060)

linear projections to the weight matrices of these layers.

- **PROMPT**: PROMPT is a Prompt-Based method, which is proposed by huang et al. (Huang et al., 2023). Specifically, the PROMPT method do not make any parameter adjustments to the Qwen2.5-Coder-3B but instead guide the Qwen2.5-Coder-3B to output contents that is free from gender bias with a meticulously designed prompt.

We use our GenBiasPro-CG to evaluate the model's degree of gender bias in relation to profession. We use the popular and classic code generation capability evaluation dataset: HumanEval (Chen et al., 2021a), HumanEval-Plus (Liu et al., 2023a), MBPP (Austin et al., 2021a), and MBPP-Plus (Liu et al., 2023a) to evaluate the model's code generation ability with the help of Pass@1 (Chen et al., 2021a). We also report the number of parameters that need to be tuned for our MSME and other baselines (abbreviated as Cost-P). The experimental results are shown in the Table 3.

From the Table 3, we can see that: (1) Except for FPFT, other baselines are not as good as our MSME method in alleviating gender bias, and models processed by these baselines are even more gender biased than the original model. Although FPFT performs well in alleviating gender bias, it requires fine-tuning all parameters, which results in high computational loss. In addition, the code generation capability of the model processed by FPFT is completely lost; (2) Our MSME approach can effectively alleviate the gender bias of the model and maintain the code generation ability of the model. For instance, by adjusting just 2.5e3 parameters in Qwen2.5-Coder-3B, we are able to reduce the gender bias in relation to profession by approximately 58%, without compromising its performance in code generation.

5.2 Verification of the Generalization of MEME

To verify the generalization of our MSME on code LLMs with different parameter sizes, we apply it to the 1.5B and 7B versions of Qwen2.5-Code, and the editing results are shown in the Table 4. From this table, we can see that the performances of Qwen2.5-Coder-1.5B and Qwen2.5-Coder-7B is consistent with that of Qwen2.5-Coder-3B. Specifically, they all effectively mitigate their degree of gender bias in relation of professionwhile ensuring their code generation capabilities. This also demonstrates the generalization of our MSME.

6 Conclusion

In this paper, we introduce a benchmark called GenBiasPro-CG and an evaluation metric named FBS to assess the extent of gender bias in code LLMs, specifically with respect to professions. Furthermore, we propose a novel model editing approach MSME. Extensive experiments demonstrate that MSME not only effectively mitigates gender bias in these models but also preserves their code generation capabilities. For instance, by adjusting just 2.5e3 parameters in Qwen2.5-Coder-3B, we are able to reduce the gender bias in relation to profession by approximately 58%, without compromising its performance in code generation. These results highlight the potential of MSME as an efficient and effective method for improving fairness in code generation models while maintaining their functionality. We hope that by mitigating the degree of gender bias in relation to profession in the code generated by code LLMs, we can reduce the gender bias in the services associated with it for people working in different professions.

7 Limitations

Due to the limitations of the previous work of Bolukbasi et al. (2016), this paper only discusses binary gender (male and female). We believe it is necessary to reiterate our position: we believe that all genders should be equal. In addition, this paper only discusses the problem of gender bias in code LLMs, but we would like to emphasize that our approach can be extended to other biases, please refer to the appendix A.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021a. Program synthesis with large language models. *Preprint*, arXiv:2108.07732.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021b. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of "bias" in NLP. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5454–5476, Online. Association for Computational Linguistics.

Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *Advances in neural information processing systems*, 29.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021b. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

David J Corliss. 2023. Designing against bias: Identifying and mitigating bias in machine learning and ai. In *Intelligent Systems Conference*, pages 411–418. Springer.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *Preprint*, arXiv:1803.03635.

Nat Friedman. 2021. Introducing github copilot: your ai pair programmer. *URL https://github. blog/2021-06-29-introducing-github-copilot-ai-pair-programmer*.

Chin-Lun Fu, Zih-Ching Chen, Yun-Ru Lee, and Hung-yi Lee. 2022. AdapterBias: Parameter-efficient token-dependent representation shift for adapters in NLP tasks. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2608–2621, Seattle, United States. Association for Computational Linguistics.

Isabel O Gallegos, Ryan A Rossi, Joe Barrow, Md Mehrab Tanjim, Sungchul Kim, Franck Dernoncourt, Tong Yu, Ruiyi Zhang, and Nesreen K Ahmed. 2024. Bias and fairness in large language models: A survey. *Computational Linguistics*, pages 1–79.

Qiuhan Gu. 2023. Llm-based code generation method for golang compiler testing. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 2201–2203.

Paula Hall and Debbie Ellis. 2023. A systematic review of socio-technical gender bias in ai algorithms. *Online Information Review*, 47(7):1264–1279.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep residual learning for image recognition. *Preprint*, arXiv:1512.03385.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. 2021. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*.

Sirui Hong, Yizhang Lin, Bangbang Liu, Binhao Wu, Danyang Li, Jiaqi Chen, Jiayi Zhang, Jinlin Wang, Lingyao Zhang, Mingchen Zhuge, et al. 2024. Data interpreter: An llm agent for data science. *arXiv preprint arXiv:2402.18679*.

Dong Huang, Qingwen Bu, Jie Zhang, Xiaofei Xie, Junjie Chen, and Heming Cui. 2023. Bias assessment and mitigation in llm-based code generation. *arXiv preprint arXiv:2309.14345*.

Tomasz Limisiewicz, David Mareček, and Tomáš Musil. 2023. Debiasing algorithm through model adaptation. *arXiv preprint arXiv:2310.18913*.

Deyuan Liu, Zhanyue Qin, Hairu Wang, Zhao Yang, Zecheng Wang, Fangying Rong, Qingbin Liu, Yanchao Hao, Bo Li, Xi Chen, Cunhang Fan, Zhao Lv, Dianhui Chu, Zhiying Tu, and Dianbo Sui. 2024a. Pruning via merging: Compressing LLMs via manifold alignment based layer merging. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17817–17829, Miami, Florida, USA. Association for Computational Linguistics.

Fang Liu, Yang Liu, Lin Shi, Houkun Huang, Ruifeng Wang, Zhen Yang, and Li Zhang. 2024b. Exploring and evaluating hallucinations in llm-powered code generation. *arXiv preprint arXiv:2404.00971*.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. 2023a. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. *Preprint*, arXiv:2305.01210.

Yan Liu, Xiaokang Chen, Yan Gao, Zhe Su, Fengji Zhang, Daoguang Zan, Jian-Guang Lou, Pin-Yu Chen, and Tsung-Yi Ho. 2023b. Uncovering and quantifying social biases in code generation. *Advances in Neural Information Processing Systems*, 36:2368–2380.

Samuel Marks, Can Rager, Eric J. Michaud, Yonatan Belinkov, David Bau, and Aaron Mueller. 2024. Sparse feature circuits: Discovering and editing interpretable causal graphs in language models. *Preprint*, arXiv:2403.19647.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2021. Fast model editing at scale. *arXiv preprint arXiv:2110.11309*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Nikita Nangia, Clara Vania, Rasika Bhalerao, and Samuel R. Bowman. 2020. CrowS-pairs: A challenge dataset for measuring social biases in masked language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1953–1967, Online. Association for Computational Linguistics.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2023. Codegen: An open large language model for code with multi-turn program synthesis. *Preprint*, arXiv:2203.13474.

nostalgebraist. interpreting gpt: the logit lens.

Danrui Qi and Jiannan Wang. 2024. Cleanagent: Automating data standardization with llm-based agents. *arXiv preprint arXiv:2403.08291*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, et al. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Chenglei Si, Yanzhe Zhang, Zhengyuan Yang, Ruibo Liu, and Diyi Yang. 2024. Design2code: How far are we from automating front-end engineering? *arXiv preprint arXiv:2403.03163*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2023. Knowledge editing for large language models: A survey. *Preprint*, arXiv:2310.16218.

Bingyang Wei. 2024. Requirements are all you need: From requirements to code with llms. *arXiv preprint arXiv:2406.10101*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Chengyue Wu, Yixiao Ge, Qiushan Guo, Jiahao Wang, Zhixuan Liang, Zeyu Lu, Ying Shan, and Ping Luo. 2024. Plot2code: A comprehensive benchmark for evaluating multi-modal large language models in code generation from scientific plots. *arXiv preprint arXiv:2405.07990*.

Zhongbin Xie and Thomas Lukasiewicz. 2023. An empirical analysis of parameter-efficient methods for debiasing pre-trained language models. *Preprint*, arXiv:2306.04067.

Jianhao Yan, Futing Wang, Yafu Li, and Yue Zhang. 2024. Potential and challenges of model editing for social debiasing. *arXiv preprint arXiv:2402.13462*.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10222–10240, Singapore. Association for Computational Linguistics.

Charles Yu, Sullam Jeoung, Anish Kasi, Pengfei Yu, and Heng Ji. 2023. Unlearning bias in language models by partitioning gradients. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6032–6048.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. 2020. Gradient surgery for multi-task learning. *Preprint*, arXiv:2001.06782.

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? *arXiv preprint arXiv:2305.12740*.

Xuanhe Zhou, Xinyang Zhao, and Guoliang Li. 2024. Llm-enhanced data management. *arXiv preprint arXiv:2402.02643*.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A The Extension of Our Work

We believe that our method can be easily extended to non binary genders, other factors that lead to gender bias, and other social biases. Taking racial bias as an example, we will explain from two aspects of dataset and MSME.

Dataset: We can use a malicious prompt to guide code LLMs to output "White", "Yellow" or "Black". Similar to gender bias in this paper, we can quantify the degree of racial bias in code LLMs by the probability of outputting "White", "Yellow" and "Black".

MSME: Our MSME can be directly transferred by simply modifying the heuristic loss to the Equation (9). However, how to determine the values of f_{White}, f_{Yellow}, and f_{Black} is also a worthwhile research question.

$$\begin{aligned}\mathcal{L}_{total} = & f_{White} \times p(\text{“White"}|pmt;\Theta) \\ & + f_{Yellow} \times p(\text{“Yellow"}|pmt;\Theta) \\ & + f_{Black} \times p(\text{“Black"}|pmt;\Theta)\end{aligned} \tag{9}$$

B Which Kind of Module is the Most Important?

From the Figure 4, we can clearly observe the coupling between different MLP modules. However, we must ask, are these three MLP modules truly identical? To answer this question, we fine-tuning parameters on individual MLP modules at layer $\mathcal{L}^{(35)}$ of Qwen2.5-Coder-3B, and the experimental results are shown in the Table 5.

From the Table 5, we can observe that: (1) Fine-tuning any individual MLP module can effectively alleviate the gender bias in relation of profession in code LLMs. (2) The model that only fine-tunes the mlp_down module outperforms the models that only fine-tune the one of other two MLP modules in mitigating gender bias in relation of profession. This also demonstrates the rationale behind our decision to only fine-tune the mlp_down module at module scale in MSME. Specifically, fine-tuning the mlp_down module alone can alleviate 67% of the original model's gender bias in relation of profession.

Table 5: The experimental results of applying MSME to individual MLP modules at layer $\mathcal{L}^{(35)}$ of Qwen2.5-Coder-3B. **Bold** indicates the best result in each column, and underline indicates the second best result.

	GenBiasPro-CG				
	GPT-Neg, FBS(↓)	**GPT-Pos, FBS(↓)**	**Comparative-Neg, FBS(↓)**	**Comparative-Pos, FBS(↓)**	**Avg., FBS(↓)**
	Original Model				
Org.	0.5864	0.5313	0.4877	0.4381	0.5109
	MSME at Module-Scale				
mlp_up	0.1849	0.2031	**0.1369**	**0.1932**	<u>0.1796</u>
mlp_gate	<u>0.1816</u>	<u>0.2015</u>	0.1515	0.2353	0.1925
mlp_down	**0.1366**	**0.1803**	<u>0.1477</u>	<u>0.2033</u>	**0.1670**