Code Defect Detection using Pre-trained Language Models with Encoder-Decoder via Line-Level Defect Localization

Jimin An[1*], YunSeok Choi[2*], Jee-Hyong Lee[1]

[1]Sungkyunkwan University, Korea
[2]Hankuk University of Foreign Studies, Korea

[*]Equal contribution

LREC-COLING 2024

Introduction

Code Defect Detection

- The process of identifying errors, bugs, or potential issues in software code.

```
1 def calc_average(arr):
2   total = 0
3   for num in arr:
4       total += num
5   return total / len(arr)
```

```
>>> calc_avg([1,2,3,4])
>>> 2.5
```

Introduction

Code Defect Detection

- The process of identifying errors, bugs, or potential issues in software code.

```
1 def calc_average(arr):
2   total = 0
3   for num in arr:
4       total += num
5   return total / len(arr)
```

```
>>> calc_avg([1,2,3,4])
>>> 2.5
>>> calc_avg([ ])
In line 5, ZeroDivisionError
```

Error

Introduction

Code Defect Detection

- The process of identifying errors, bugs, or potential issues in software code.
- Identifying these defects is important to improve the quality of the overall SW.

```
1 def calc_average(arr):
2   total = 0
3   for num in arr:
4       total += num
5   return total / len(arr)
```

Defect

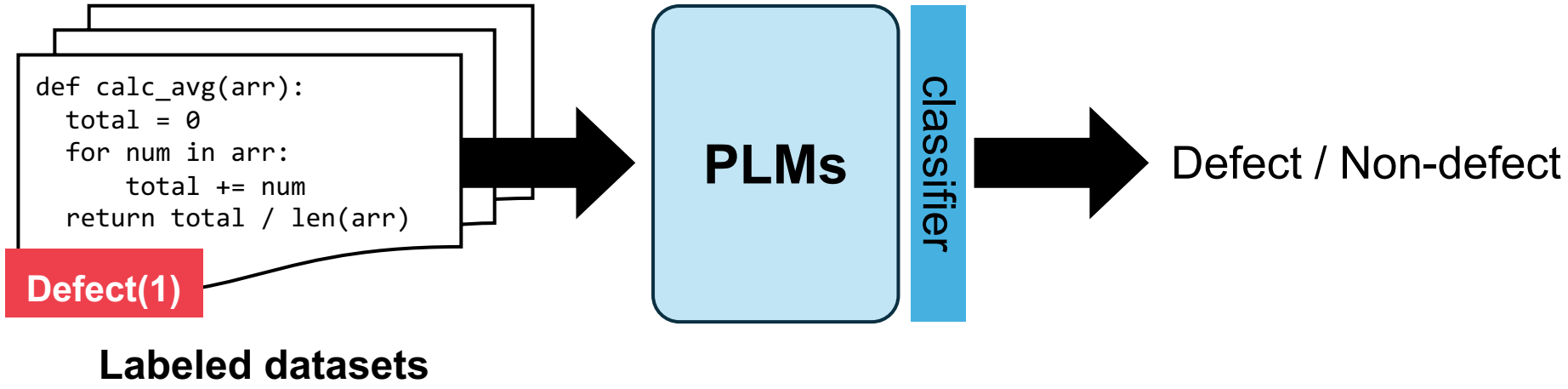
```
>>> calc_avg([1,2,3,4])
>>> 2.5
>>> calc_avg([ ])
In line 5, ZeroDivisionError
```

Error

Introduction

Defect Detection on Pre-trained Language Models(PLMs)

- Fine-tuned on labeled datasets where code are tagged as either containing defects or not.
- Most PLMs focused solely on classifying whether the code has defects or not.



Introduction

Defect Detection on Pre-trained Language Models(PLMs)

- Fine-tuned on labeled datasets where code are tagged as either containing defects or not.
- Most PLMs focused solely on classifying whether code has defects or not.

Need a more analyzable and Explainable Approach !
→ **Line-Level Defect Localization**

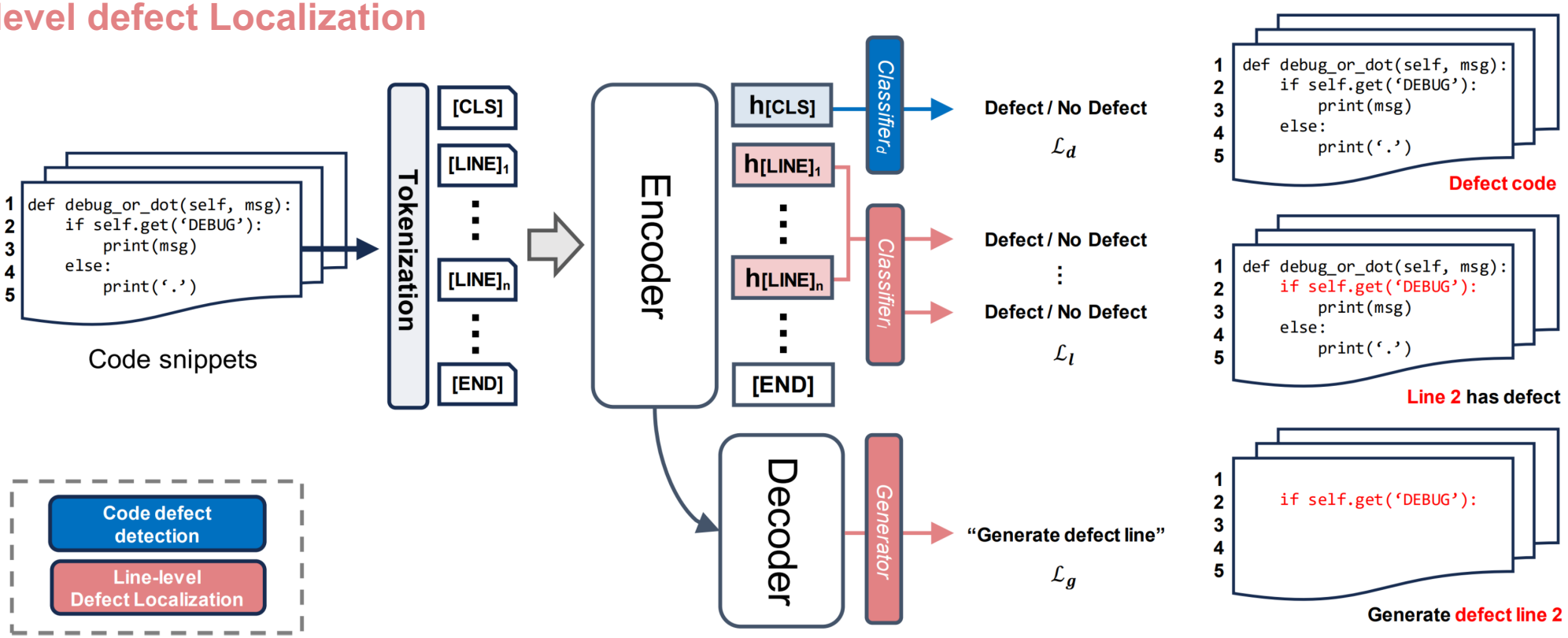
Proposed Methods

Overview of Proposed Methods

Utilizes Encoder and Decoder of PLMs for code defect detection

① **Code defect detection**

② **Line-level defect Localization**



Proposed Methods

① Code defect detection

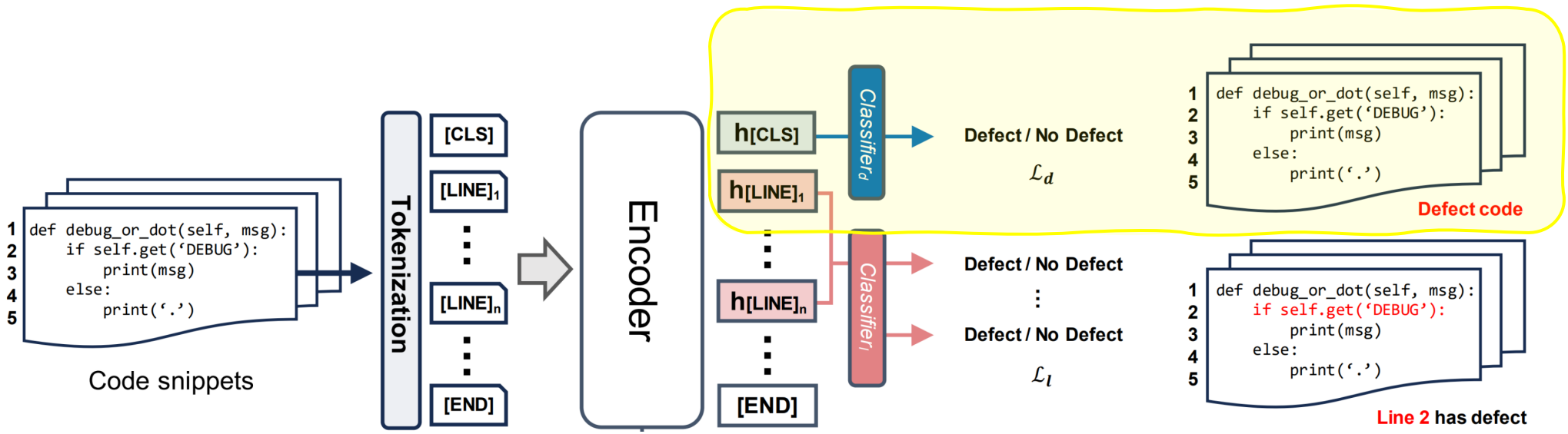
Use the encoder of PLMs with the primary goal of classifying code defect detection.

$$I = \langle [\text{CLS}]; \text{C}; [\text{END}] \rangle \quad (1)$$

$$h = \text{Encoder}(I) \quad (2)$$

$$logit_d = \text{Classifier}_d(h_{[\text{CLS}]}) \quad (3)$$

$$\mathcal{L}_d = \text{CrossEntropy}(logit_d, label_d) \quad (4)$$



Proposed Methods

② Line-level defect localization

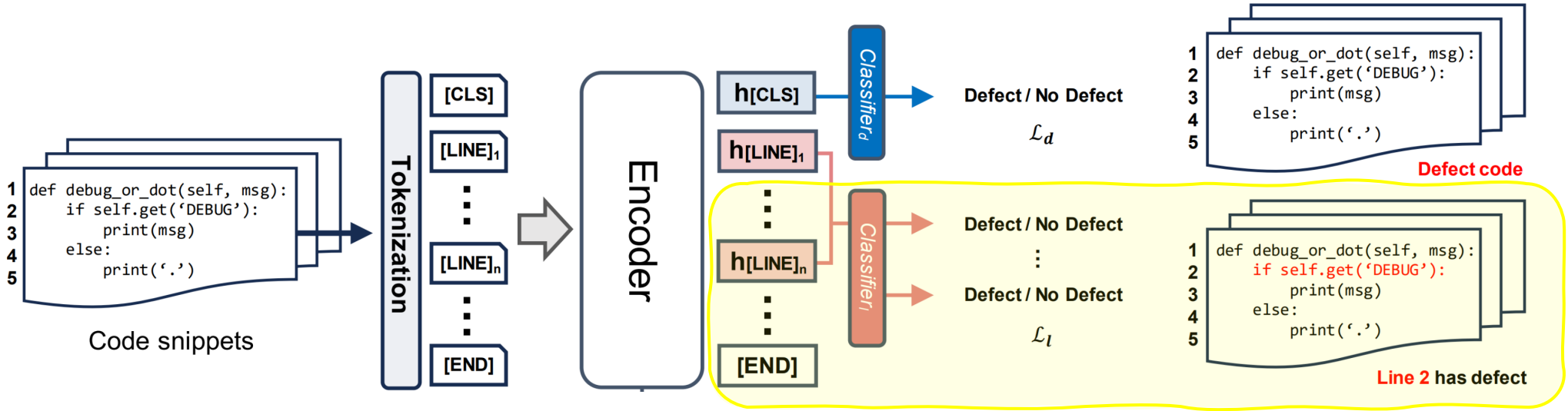
Identifying the specific locations of defects within the code, classifying defective lines on a line-by-line basis.

$$I = \langle [\text{CLS}]; [\text{LINE}]_i; c_1; ...; [\text{LINE}]_n; c_n; [\text{END}] \rangle \quad (5)$$

$$h = \text{Encoder}(I) \quad (6)$$

$$logit_{l_i} = \text{Classifier}_l(h_{[\text{LINE}]_i}) \quad (7)$$

$$\mathcal{L}_l = \sum_{i=1}^{n} \text{CrossEntropy}(logit_{l_i}, label_{l_i}) \quad (8)$$



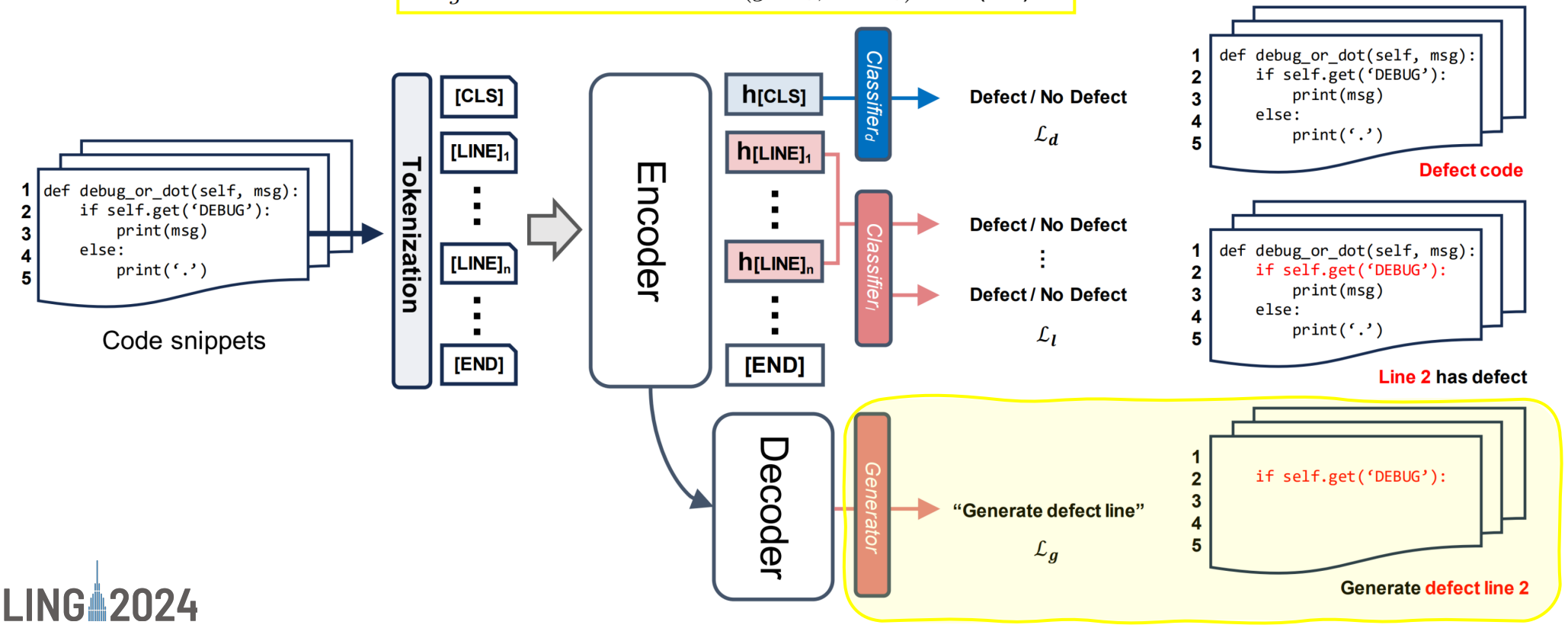
Proposed Methods

② Line-level defect localization

Conduct the task of generating defective lines in the decoder for line-level localization.

$$gen_i = \text{Decoder}(h) \quad (9)$$

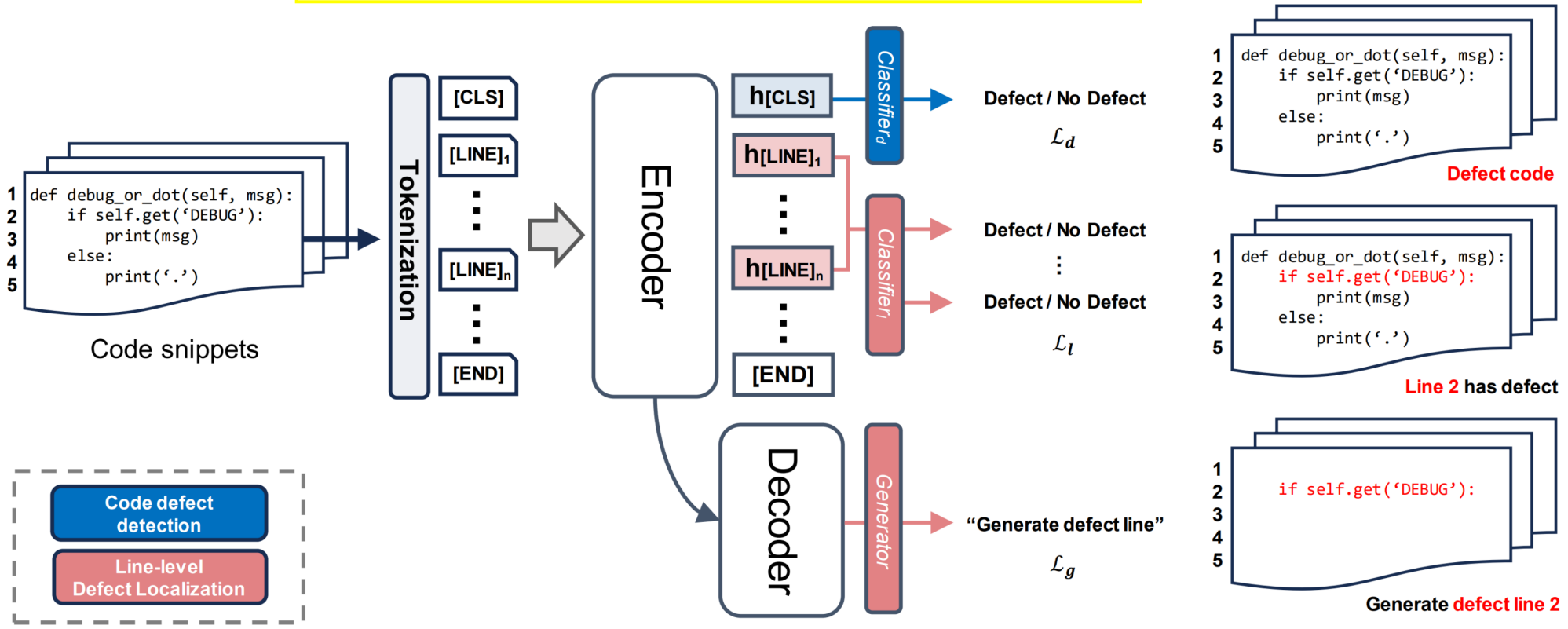
$$\mathcal{L}_g = \text{GenerationLoss}(gen_i, sent_i) \quad (10)$$



Proposed Methods

Unified Multi-task Training

$$\mathcal{L}_{final} = w_1 * \mathcal{L}_d + w_2 * \mathcal{L}_l + w_3 * \mathcal{L}_g \qquad (11)$$



Experiments Setup

4 benchmark datasets

- Devign
- Variable-Misuse(VM)
- Wrong Binary Operator(WBO)
- Swapped Operand(SO)

Baseline

- 3 Encoder Decoder-based PLMs: CodeT5, CodeT5+, UniXcoder
- 2 Encoder-based PLMs: CodeBERT, CuBERT

Main Result

Comparison of our proposed method with the baseline models

Datasets	Devign		VM		WBO		SO	
Models	Acc.	F1.	Acc.	F1.	Acc.	F1.	Acc.	F1.
CuBERT (Kanade et al., 2020)	-	-	94.04	-	89.90	-	92.20	-
CodeBERT (Feng et al., 2020a)	63.73	51.51	93.21	93.03	90.66	90.27	91.06	90.77
CodeT5 (Wang et al., 2021a)	62.87	58.39	93.82	93.74	88.12	87.75	91.78	91.70
CodeT5+ (Wang et al., 2023)	63.40	62.59	93.28	93.21	89.08	88.62	92.70	92.61
UniXcoder (Ahmad et al., 2021)	63.18	47.57	93.95	93.85	90.35	90.11	93.73	93.66
CodeT5 (ours)	**65.44**	62.68	**95.43**	**95.38**	90.53	90.29	93.88	93.80
CodeT5+ (ours)	64.91	**63.97**	95.08	95.02	91.56	91.35	93.69	93.62
UniXcoder (ours)	64.29	58.37	95.36	95.30	**92.49**	**92.31**	**94.22**	**94.16**

Ablation Study

Three learning processes of our proposed method

(1) Code Defect detection in Encoder: $\mathcal{L}_d = \text{CrossEntropy}(logit_d, label_d)$

(2) Line-level Defect Localization in Encoder: $\mathcal{L}_l = \sum_{i=1}^{n} \text{CrossEntropy}(logit_{l_i}, label_{l_i})$

(3) Line-level Defect Localization in Decoder: $\mathcal{L}_g = \text{GenerationLoss}(gen_i, sent_i)$

	Devign		VM		WBO		SO	
	Acc.	F1.	Acc.	F1.	Acc.	F1.	Acc.	F1.
Baseline: (1)	62.87	58.39	93.82	93.74	88.12	87.75	91.78	91.70
Ours: (1)+(2)	63.18	<u>58.88</u>	<u>94.76</u>	<u>94.70</u>	89.13	88.84	92.35	92.28
Ours: (1)+(3)	<u>63.67</u>	56.01	94.12	94.06	<u>89.97</u>	<u>89.68</u>	<u>92.80</u>	<u>92.72</u>
Ours: (1)+(2)+(3)	**65.44**	**62.68**	**95.43**	**95.38**	**90.53**	**90.29**	**93.88**	**93.80**

* We chose CodeT5 as our baseline in this table.

Conclusion and Future Work

Conclusion

- We introduced a novel method for code defect detection with line-level defect localization in a unified manner.

- By segmenting the code based on lines and leveraging both the encoder and decoder of PLMs, we achieved a more detailed and interpretable defect detection mechanism.

Future work

- We plan to conduct research on an integrated model that simultaneously performs code defect detection and defect repair based on line-level defect information.

Thank you:)