Paper ID: 194

UnetPro: Combining Attention with Skip Connection in Unet for Efficient IR Drop Prediction

Zhengfei Qi[1], Wanchao Wang[1], Chengxuan Yu[1], Dan Niu[2], Xiao Wu[3], Zhou Jin[1]

1. Super Scientific Software Laboratory, China University of Petroleum, Beijing

2. School of Automation, Southeast University

3. Huada Empyrean Software Co. Ltd

Email: zhengfei.qi@student.cup.edu.cn

Background

What is IR Drop?

IR drop is the voltage reduction in an integrated circuit caused by current flowing through wires and resistive elements.

Why is IR Drop important?

It's important because it can cause different circuit parts to operate at varying voltage levels, affecting overall performance.

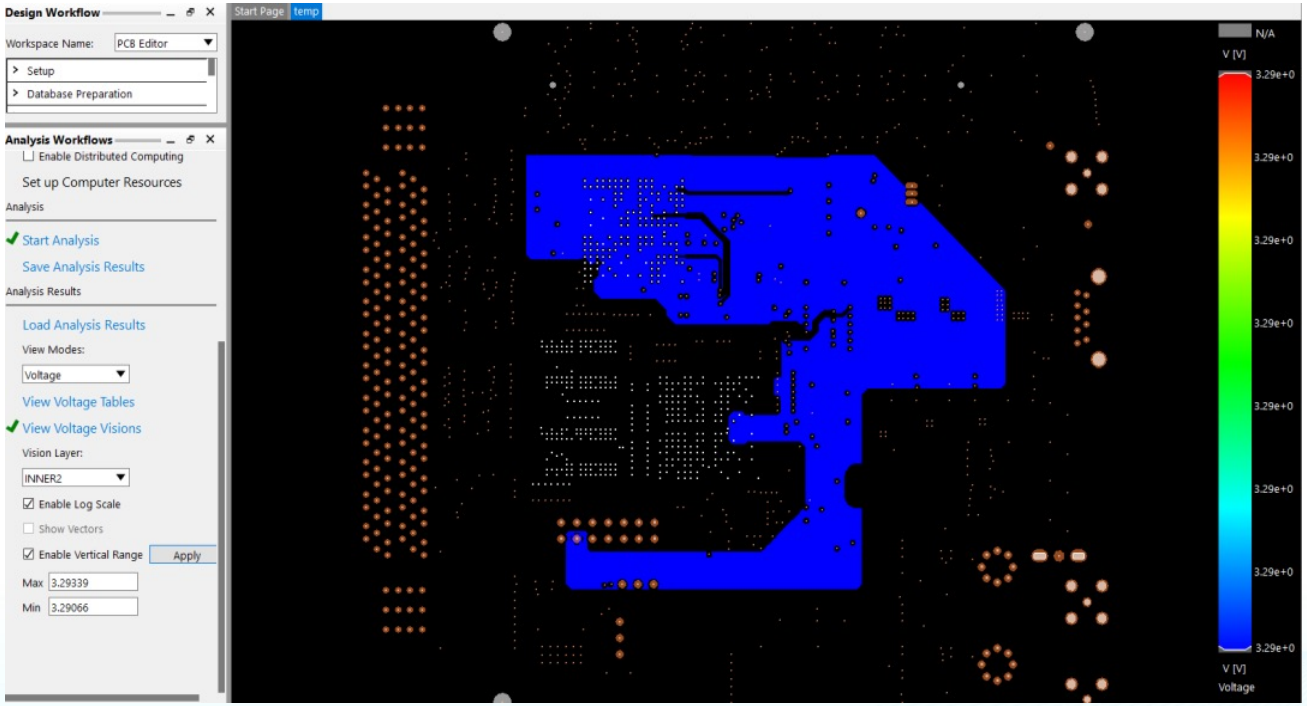


Figure 1: The IR Drop Simulation.

Background

Traditional IR drop analysis relies on mathematical methods, which can be time-consuming due to solving large matrices. The rise of machine learning has brought new approaches to IR drop prediction. **Machine learning models can learn from a large amount of circuit design data and actual performance information to achieve accurate prediction of circuit IR drop.**

As shown in Figure 2, the Power Delivery Network (PDN) is subjected to feature extraction and transformed into a IR drop distribution map by convolutional neural network.

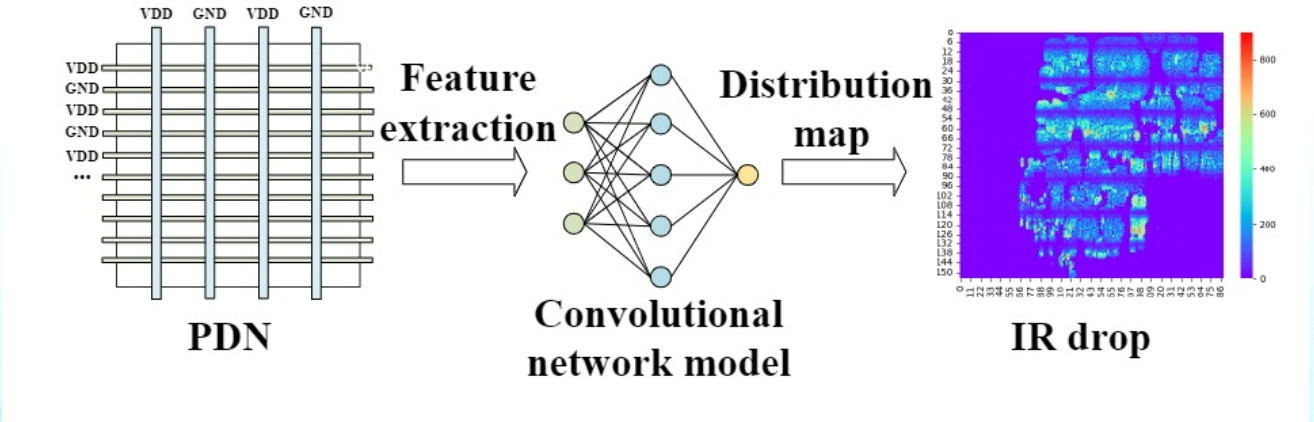


Figure 2: The whole framework.

Background

Some previous works extract localized properties of PDN into XGBoost [1] to predict the IR drop.

problem

But may be difficult to apply to whole chip analysis.

Some works consider the power transfer noise map as an image and employ a CNN-based [2] [3] [4] strategy for IR drop prediction.

problem

But may have problems with prediction accuracy.

[1] C.-H. Pao, A.-Y. Su, and Y.-M. Lee. Xgbir: An xgboost-based ir drop predictor for power delivery network. In DATE ’20, 2020.
[2] C.-T. Ho and A. B. Kahng. Incpird: Fast learning-based prediction of incremental ir drop. In ICCAD ’19, 2019.
[3] Z. Xie, H. Ren, B. Khailany, Y. Sheng, S. Santosh, J. Hu, and Y. Chen. Powernet: Transferable dynamic ir drop estimation via maximum convolutional neural network. In ASP-DAC ’20,2020.
[4] X. X. Huang, H. C. Chen, S. W. Wang, I. H. R. Jiang, Y. C. Chou, and C. H. Tsai. Dynamic ir-drop eco optimization by cell movement with current waveform staggering and machine learning guidance. In ICCAD’20, 2020.

Background

Challenge :

- Lack of data or imbalance in data distribution may hinder the model from fully learning relevant features. This makes **feature extraction** very challenging.
- As the number of network layers increases, there is more **information loss.**

Contributions :

- We introduce **multi-scale convolution** and **attention**. This enhances the feature extraction function of the model.
- Our model integrates **skip connection** to reduce loss of information transmission.

Background

Our overall process is shown in Figure 3. It mainly includes the following parts:

- **Data pre-processing**: Feature selection、Data-enhancement、Data segmentation
- **Model construction**: Multi-scale Convolution、UnetPro Framework
- **Model evaluation and validation**

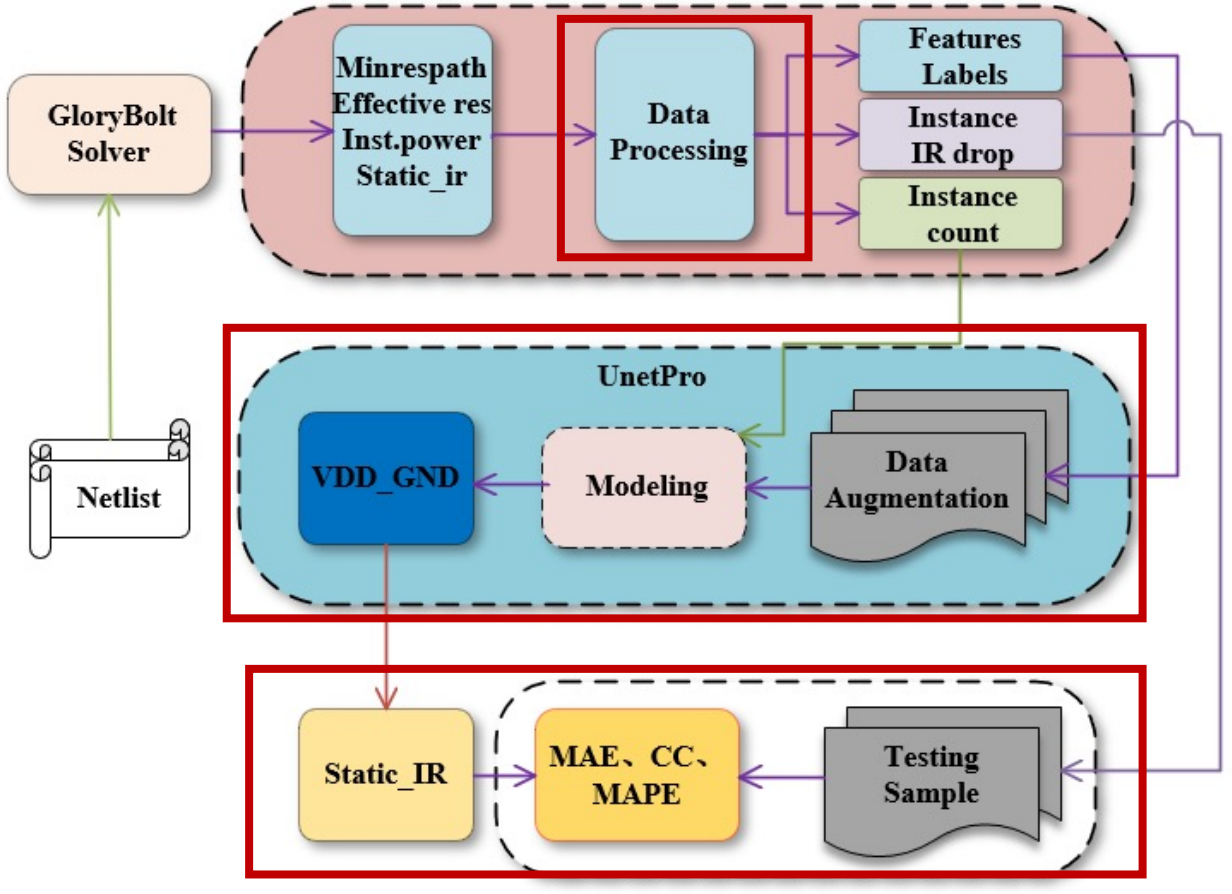


Figure 3: The flow chart.

Data-preprocessing

- Feature selection

As shown in the figure on the right, we select three features from the raw data, as well as two labeled data.

- **total_power**: Total power consumption of the instance
- **eff_VDD**: The equivalent resistance of the instance to the power net
- **eff_VSS**: Equivalent resistance of instance to ground net
- **VDD_drop**: IR drop on instance power pin
- **GND_bounce**: ground bounce on instance ground pin

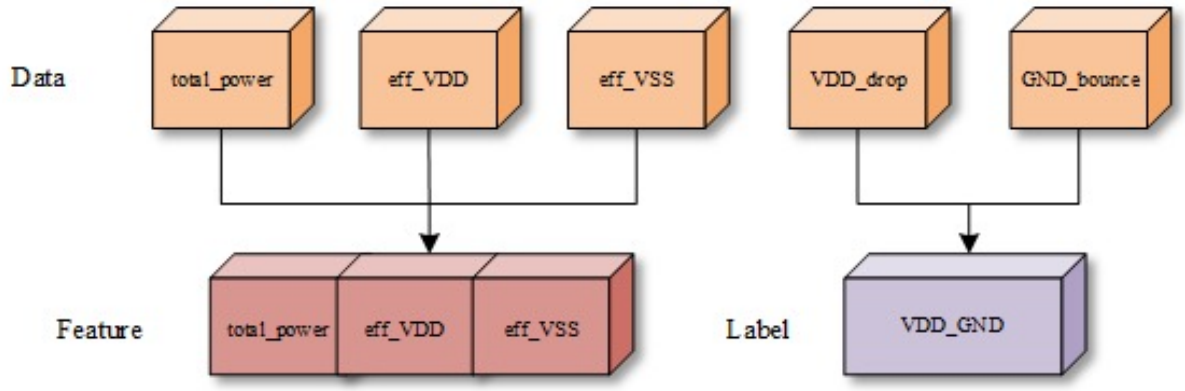


Figure 4: Feature selection.

Additionally, we notice that the label dimension may affect the model training speed and prediction performance. Therefore, we reduce the prediction to one dimension by **summing the values of VDD and GND**.

Data-preprocessing

- Data-enhancement

In order to improve the model performance, we introduce data augmentation techniques, which use three main data augmentation operations, i.e., **Flip**, **Rotation**, and **Crop**, to increase the diversity of the training data, so that the model can better adapt to different input conditions.

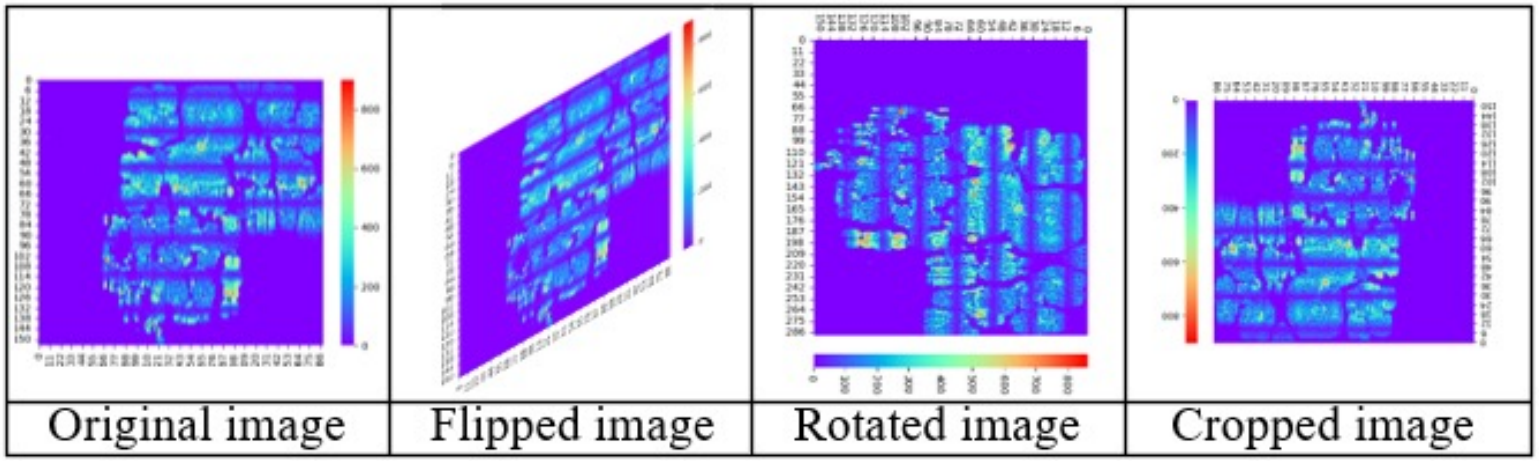


Figure 5: Data enhancement diagram.

Data-preprocessing

- Data segmentation

Table 1: Data set.

DataSet	Nvdla-small	RISCY	RISCY-FPU	Zero-riscy	Vortex-small
Number	226*1	150*16	150*16	150*16	66*1

Figure 6: Data segmentation diagram.

There are two groups of data with relatively smaller sample sizes.

To ensure a certain level of performance for these small-sample data types and fully utilize all available data, we choose a **rolling-learning-like** approach. Each round incorporates the previous test set into the next training set, ensuring optimal use of available data.

Algorithm

- Multi-scale Convolution

In order to further enhance the feature extraction performance of the model, we use multi-scale convolution module with **different sizes of convolution kernels** on the data for extraction.

It consists of multiple **parallel** convolutional layers, each of which captures different scale features of the input data. Following capture, the output from these layers is fused for final output.

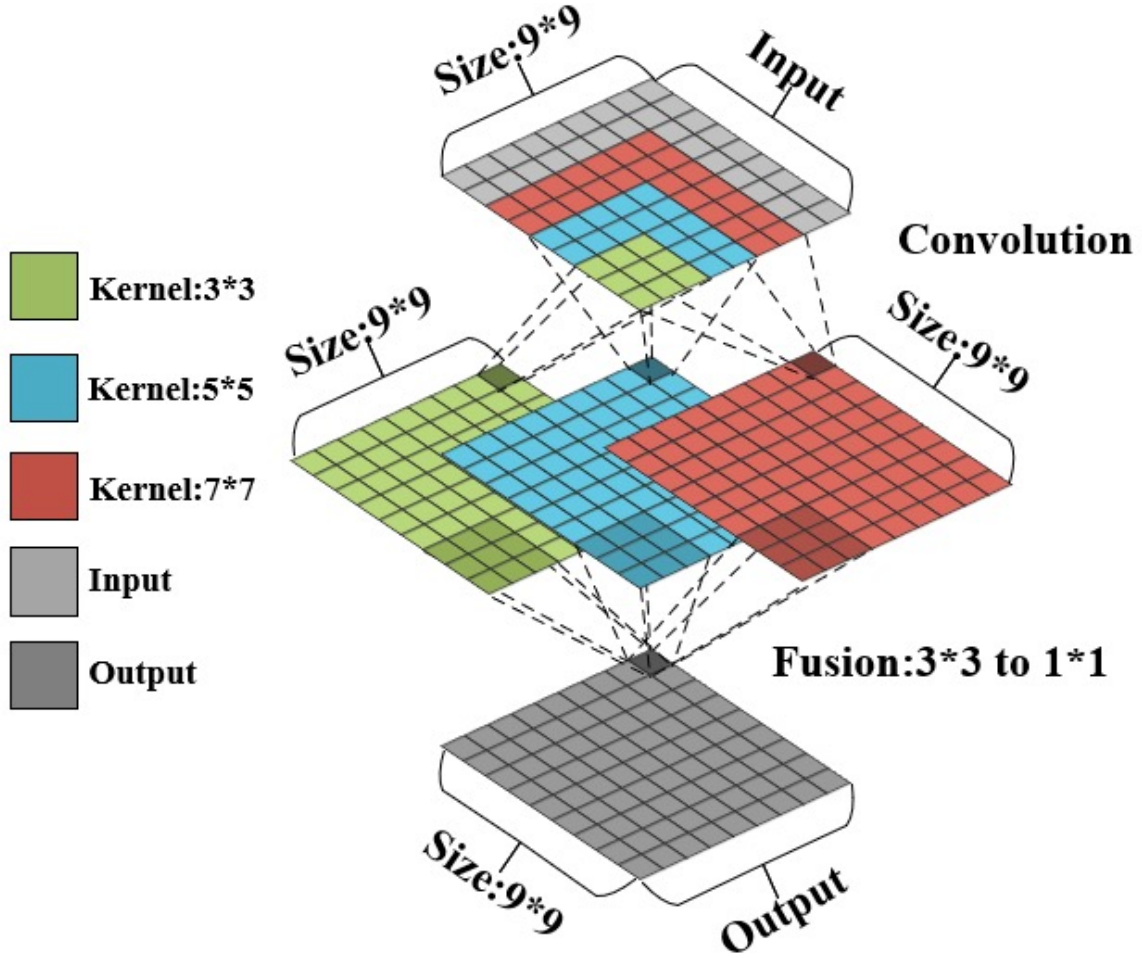


Figure 7: Data segmentation diagram.

Algorithm

- UnetPro Framework

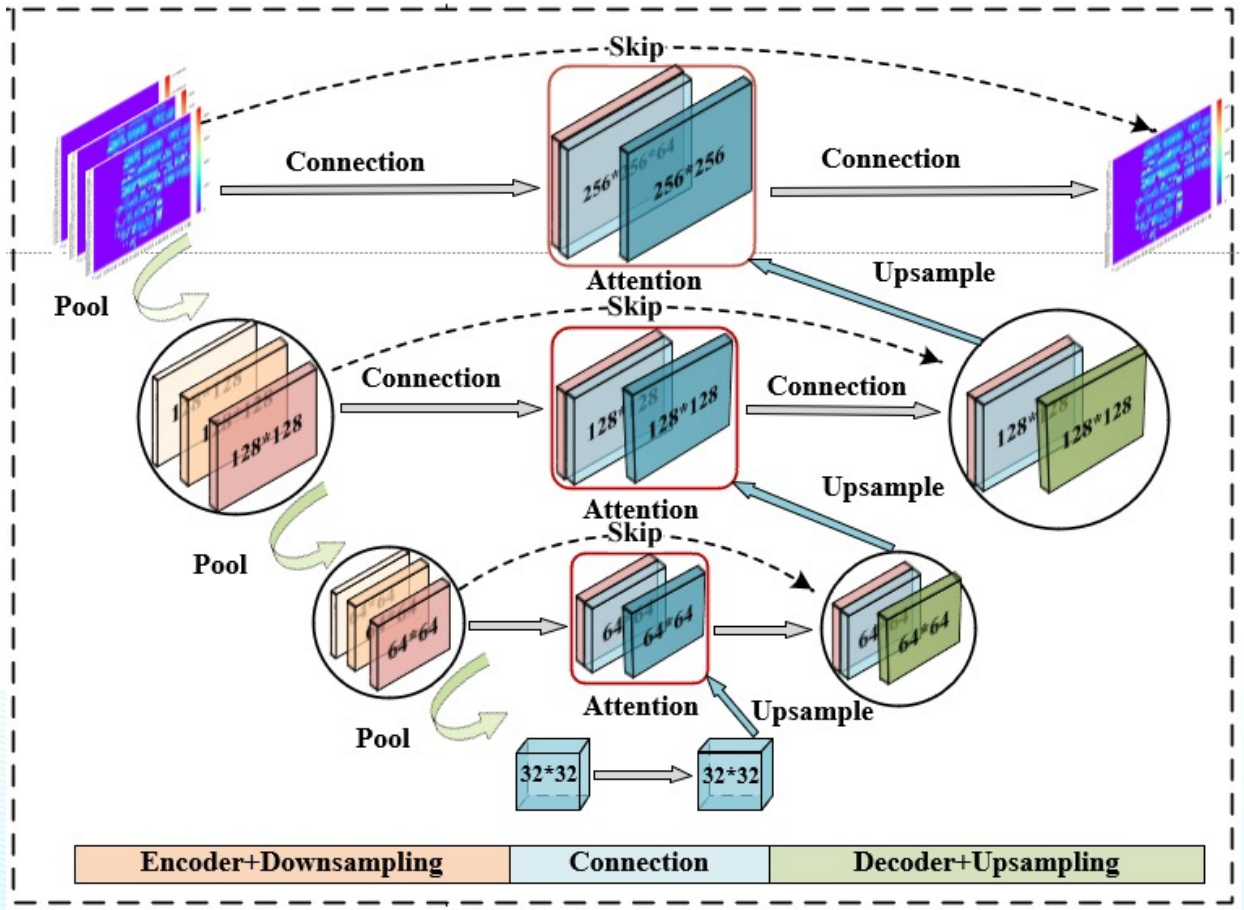


Figure 8: Data segmentation diagram.

Our model is divided into two parts: an encoder on the left and a decoder on the right, with a total of four layers from top to bottom.

Algorithm

- UnetPro Framework

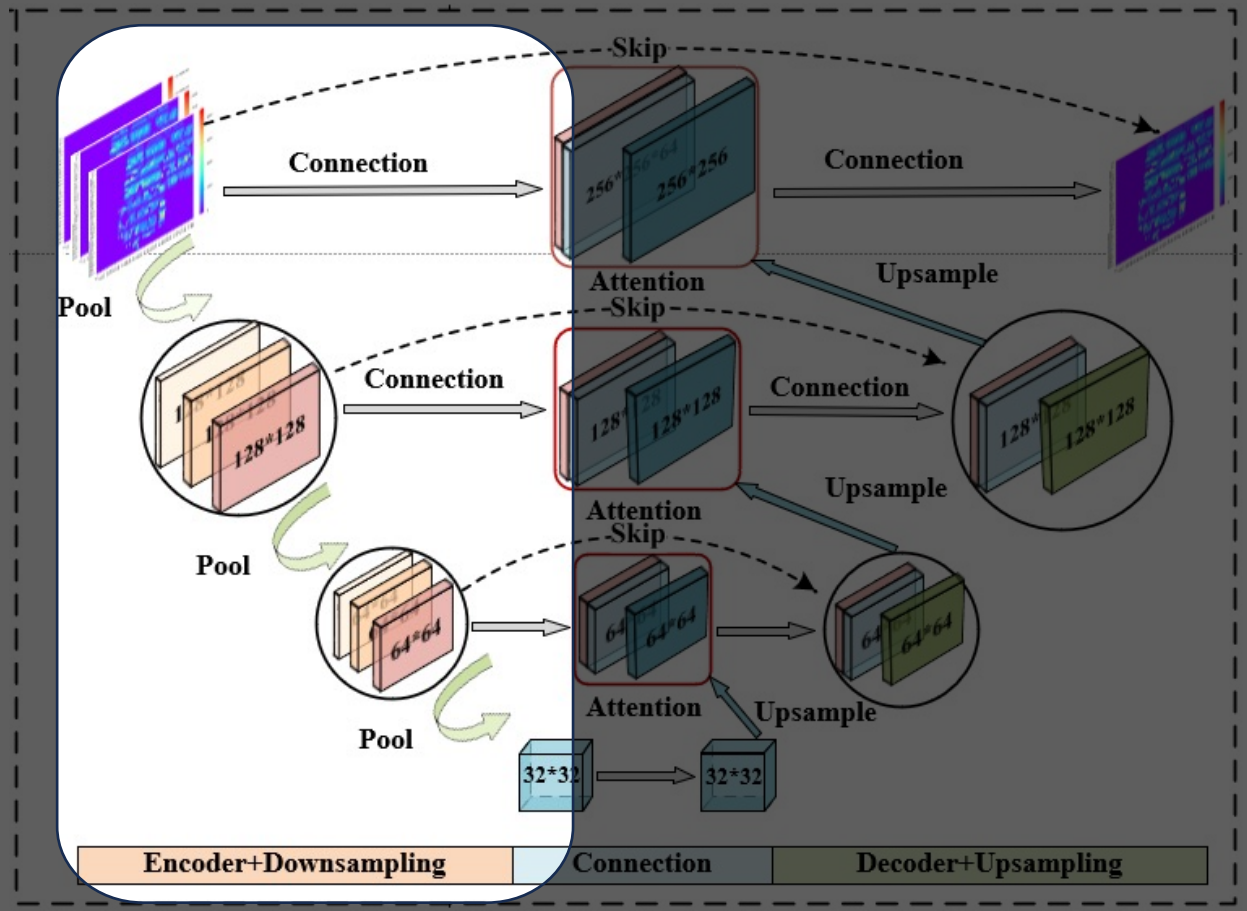


Figure 7: Data segmentation diagram.

Our model is divided into two parts: an **encoder** on the left and a decoder on the right, with a total of four layers from top to bottom.

Algorithm

- UnetPro Framework

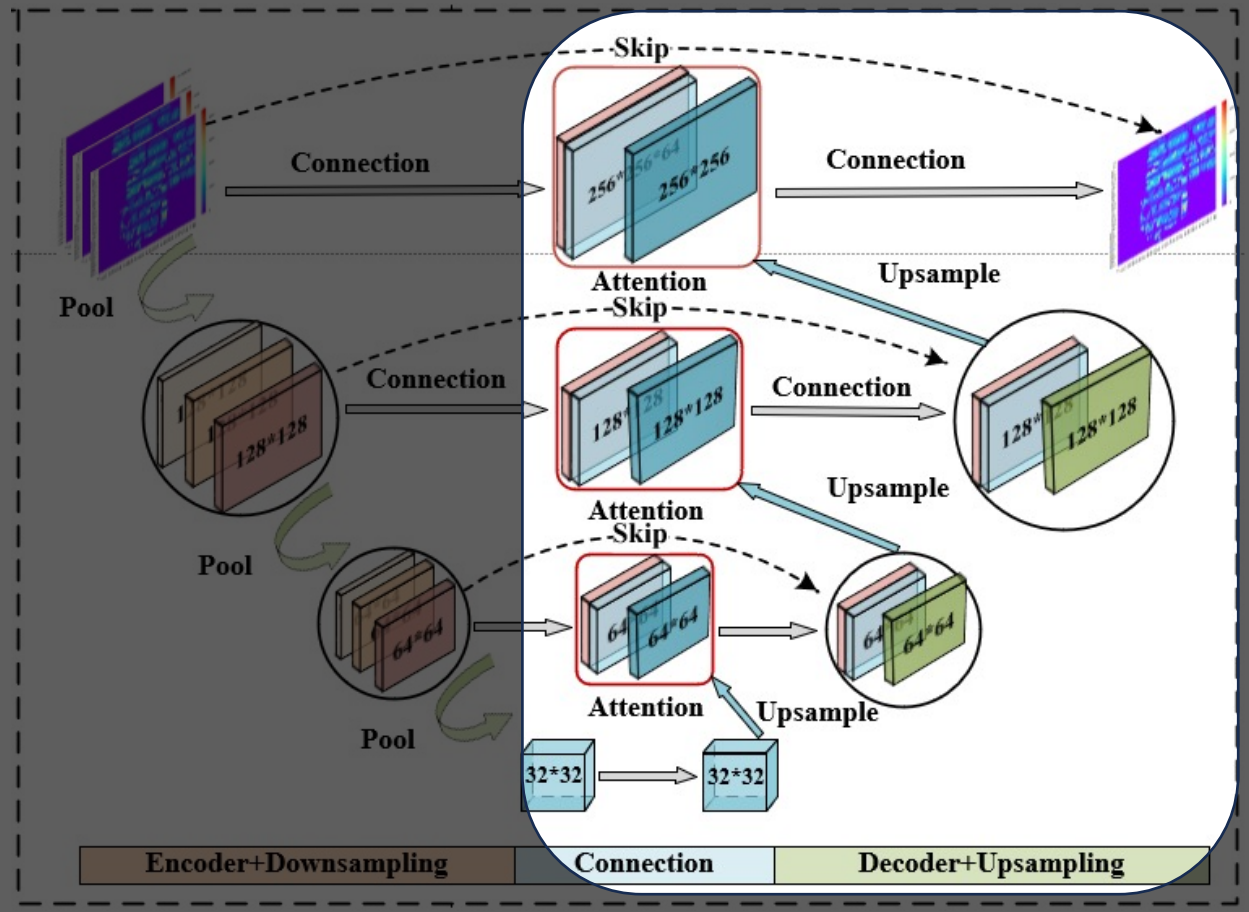


Figure 7: Data segmentation diagram.

Our model is divided into two parts: an encoder on the left and a **decoder** on the right, with a total of four layers from top to bottom.

Algorithm

- UnetPro Framework

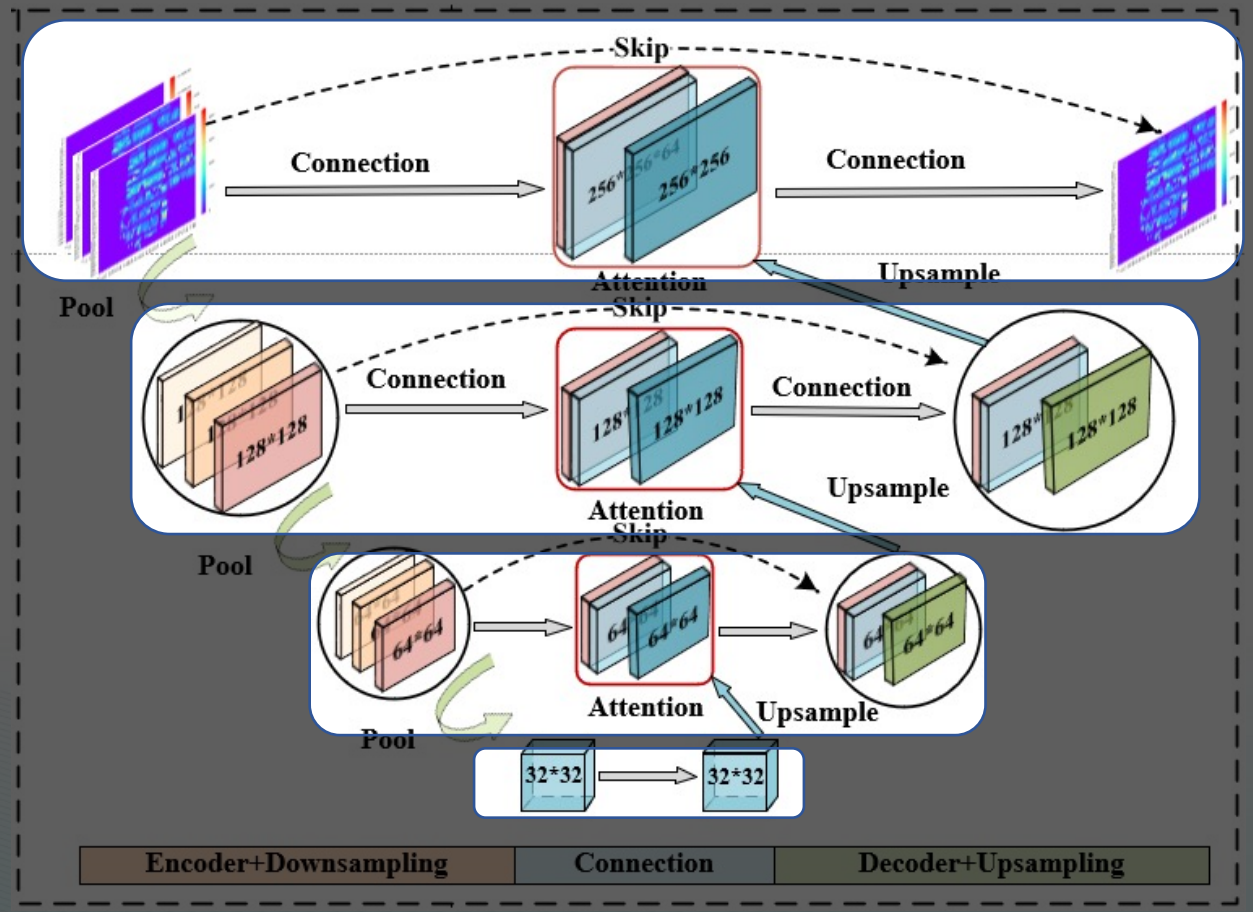


Figure 7: Data segmentation diagram.

Our model is divided into two parts: an encoder on the left and a decoder on the right, with a total of **four layers** from top to bottom.

Algorithm

- UnetPro Framework

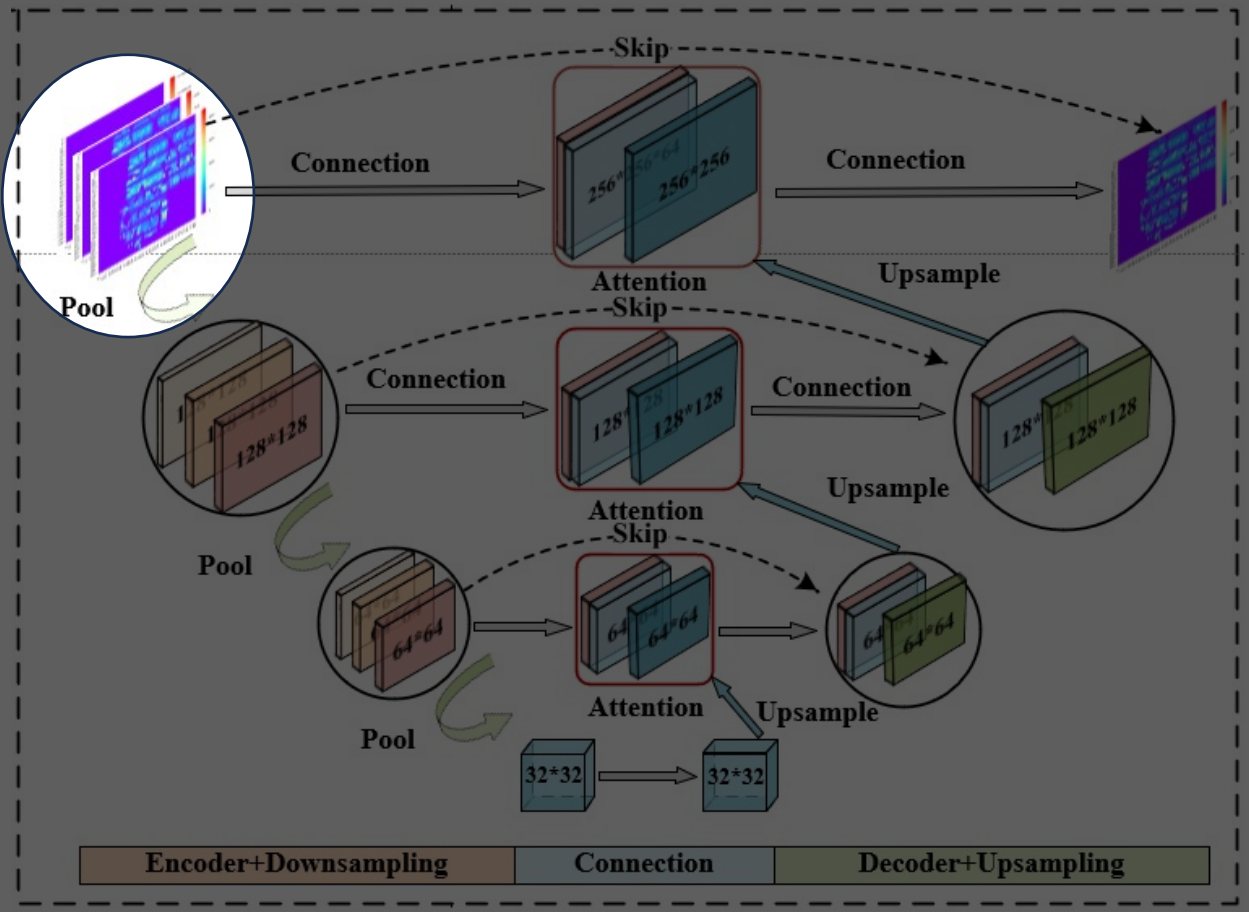


Figure 7: Data segmentation diagram.

- After the input of image data into the encoder, it undergoes the first layer with **two convolutional structures.**
- Then it proceeds to the next layer for **pooling**, and so on.
- When reaching the fourth layer, the encoder outputs the results of each layer to the decoder.

Algorithm

- UnetPro Framework

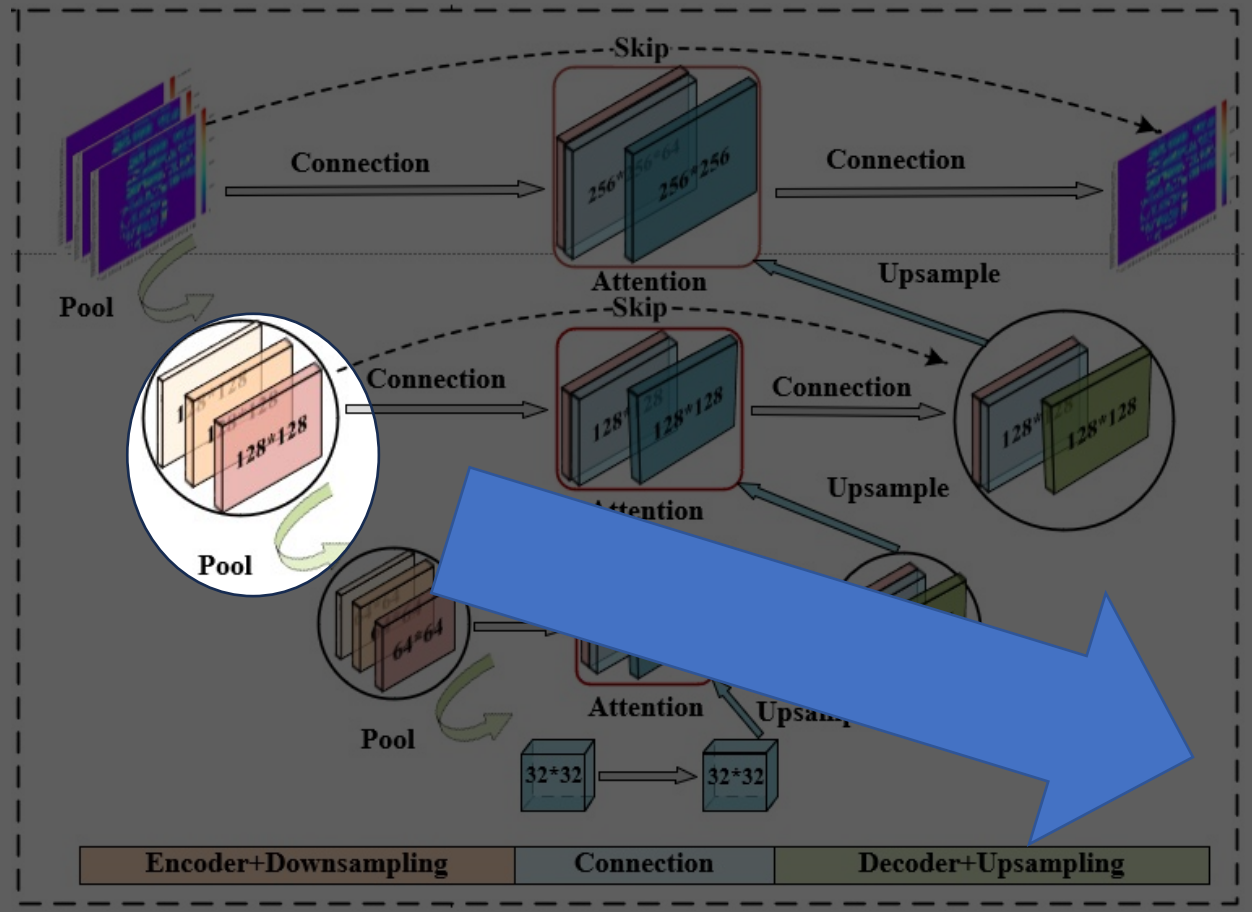
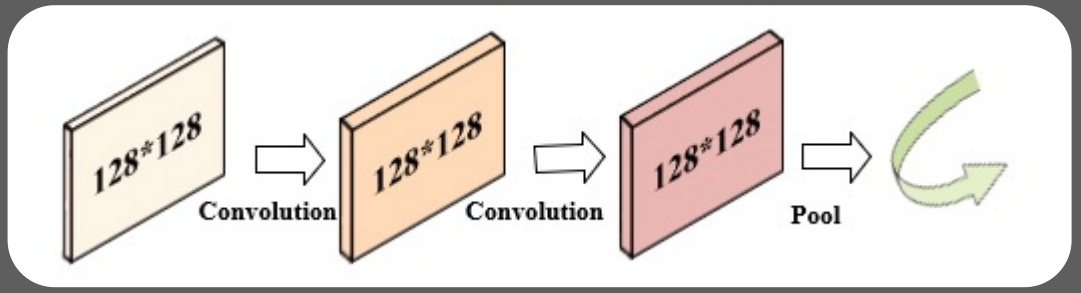


Figure 7: Data segmentation diagram.

- After the input of image data into the encoder, it undergoes the first layer with **two convolutional structures**.
- Then it proceeds to the next layer for **pooling**, and so on.
- When reaching the fourth layer, the encoder outputs the results of each layer to the decoder.



Algorithm

- UnetPro Framework

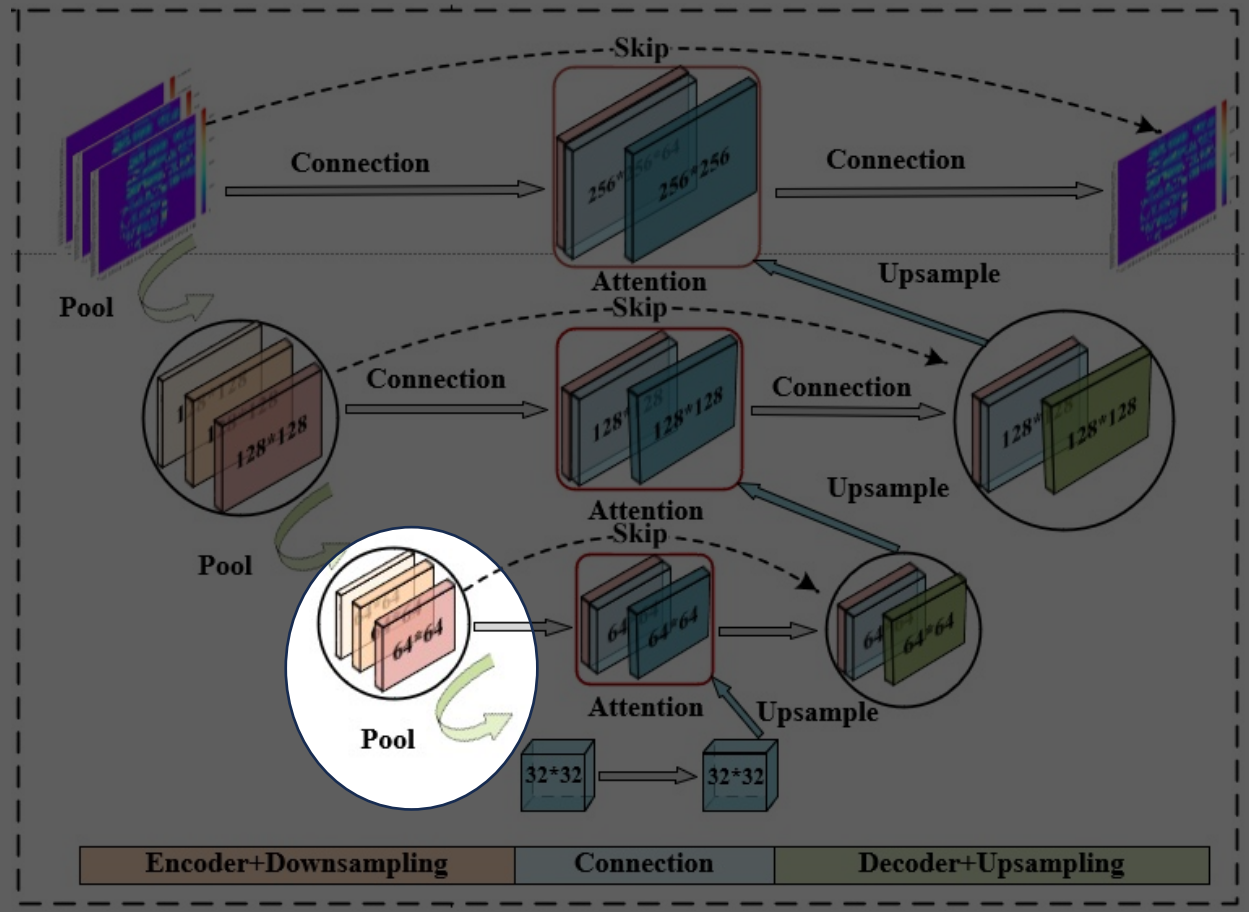


Figure 7: Data segmentation diagram.

- After the input of image data into the encoder, it undergoes the first layer with **two convolutional structures**.
- Then it proceeds to the next layer for **pooling**, and so on.
- When reaching the fourth layer, the encoder outputs the results of each layer to the decoder.

Algorithm

- UnetPro Framework

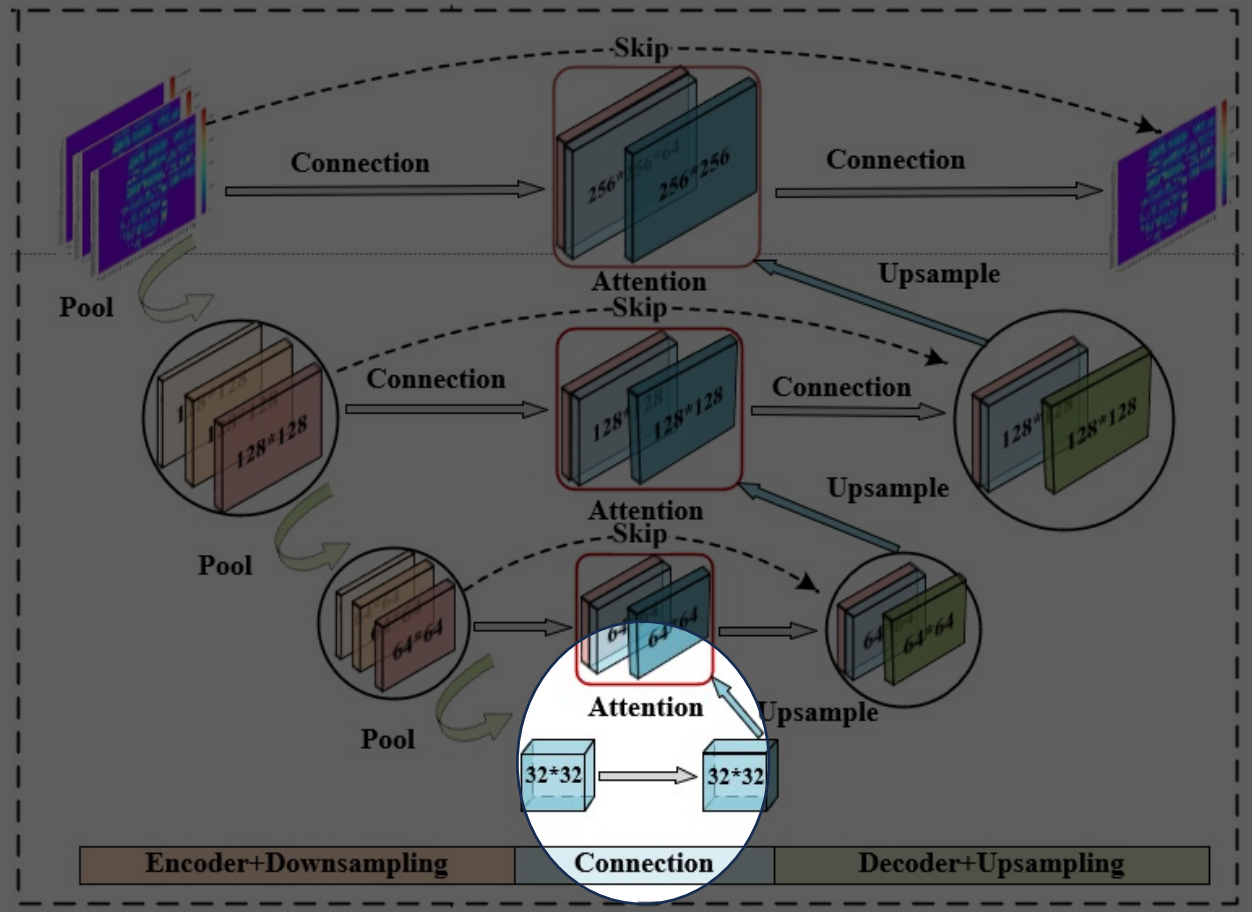


Figure 7: Data segmentation diagram.

- After the input of image data into the encoder, it undergoes the first layer with **two convolutional structures**.
- Then it proceeds to the next layer for **pooling**, and so on.
- When reaching the fourth layer, the encoder outputs the results of each layer to the decoder.

Algorithm

- UnetPro Framework

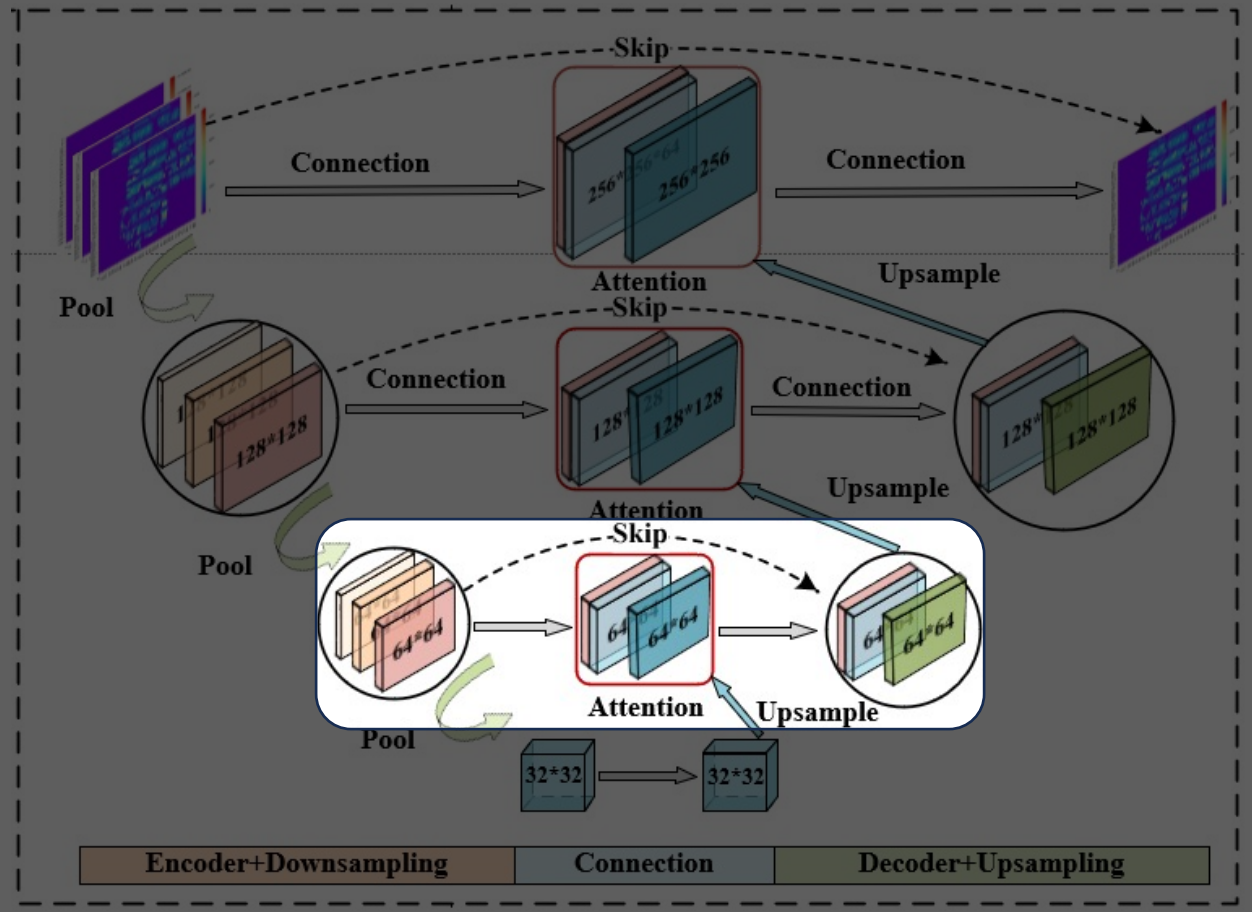


Figure 7: Data segmentation diagram.

- The decoder first performs **upsampling** on the results from the encoder, and then **concatenates** it with the results from the third layer.
- The combined output then passes through an **attention module** to obtain the result.
- This result is **attention module** with the output of the third layer and pass to the next layer.

Algorithm

- UnetPro Framework

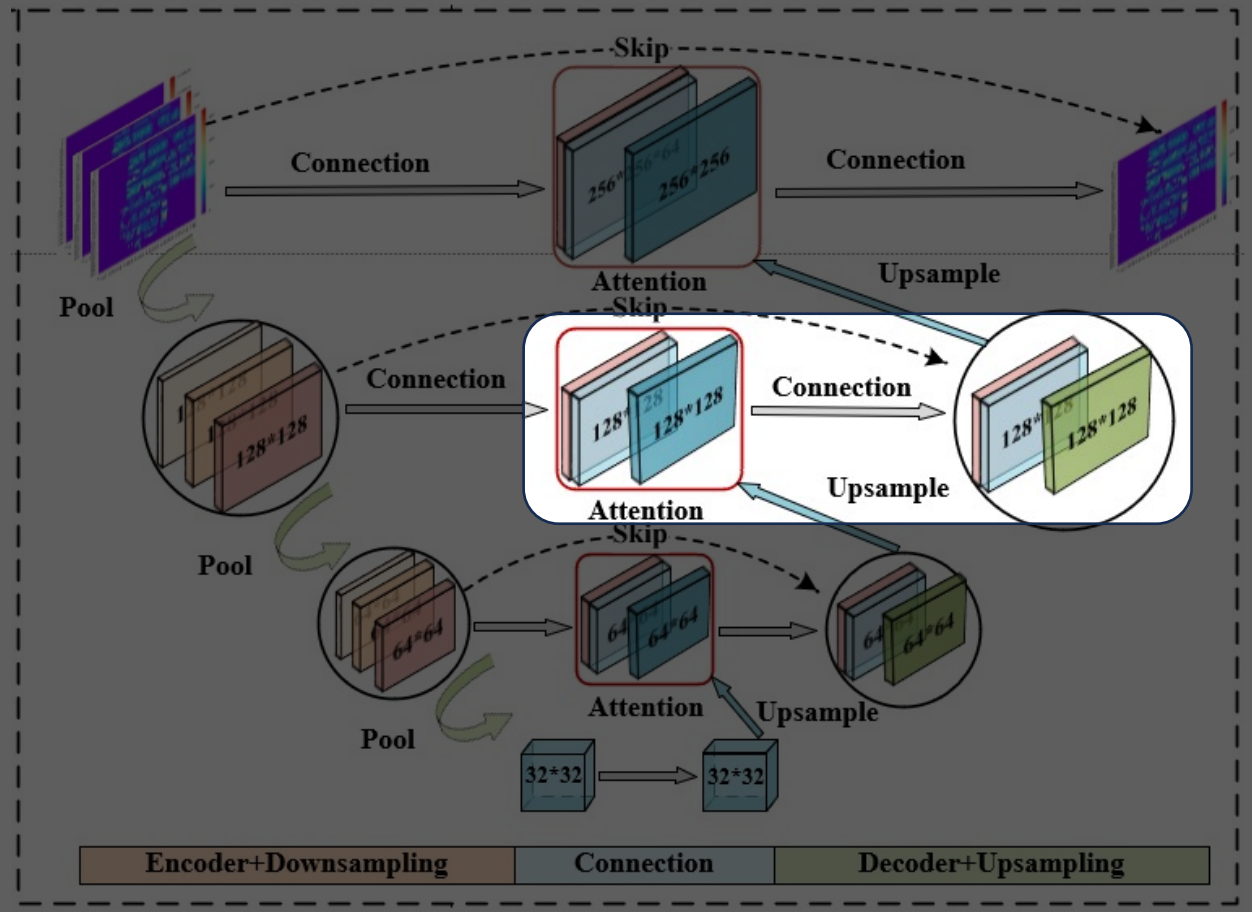


Figure 7: Data segmentation diagram.

- The decoder first performs **upsampling** on the results from the encoder, and then **concatenates** it with the results from the third layer.
- The combined output then passes through an **attention module** to obtain the result.
- This result is **attention module** with the output of the third layer and pass to the next layer.

Algorithm

- UnetPro Framework

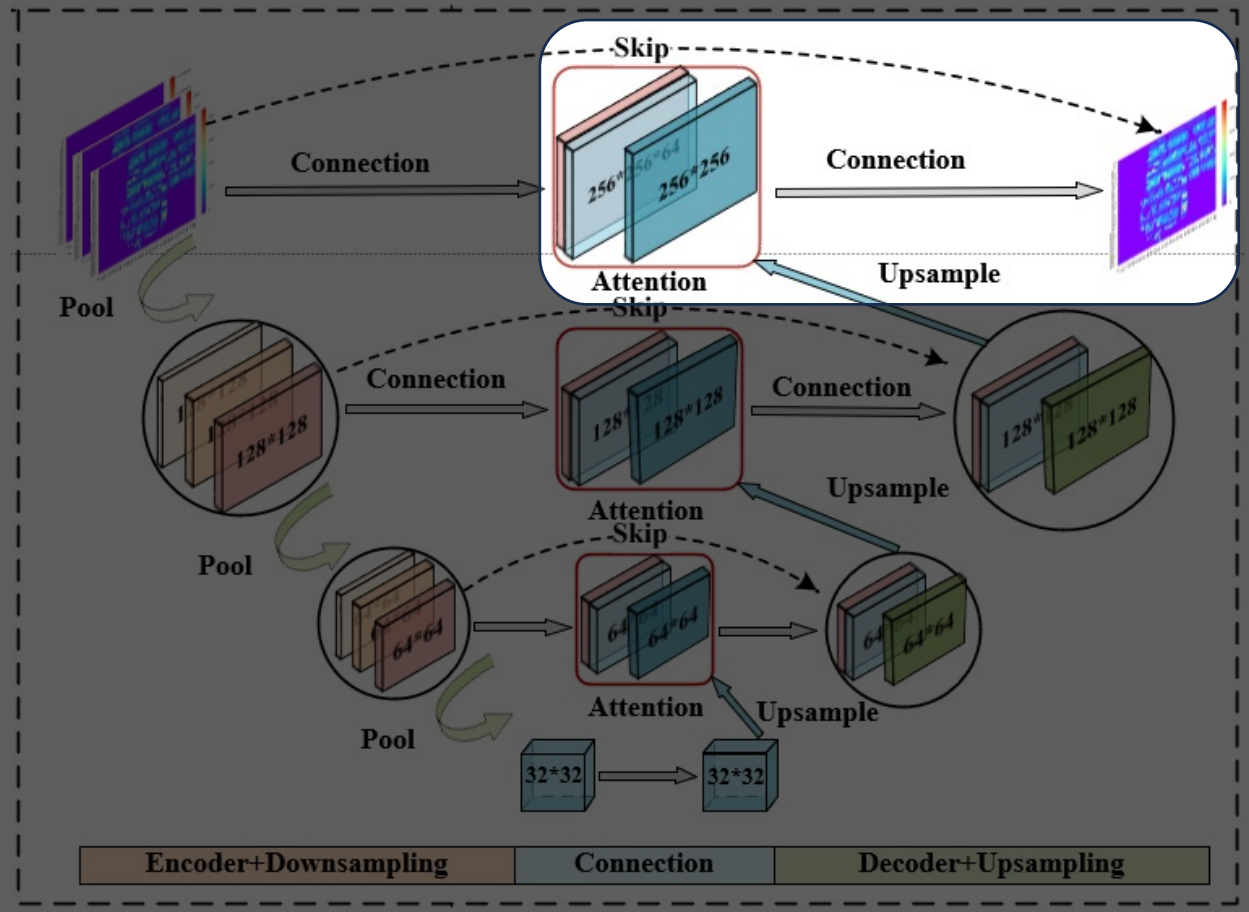


Figure 7: Data segmentation diagram.

- The decoder first performs **upsampling** on the results from the encoder, and then **concatenates** it with the results from the third layer.
- The combined output then passes through an **attention module** to obtain the result.
- This result is **attention module** with the output of the third layer and pass to the next layer.

Experiment Setup

Table 1: Data set.

DataSet	Nvdla-small	RISCY	RISCY-FPU	Zero-riscy	Vortex-small
Number	226*1	150*16	150*16	150*16	66*1

- Our dataset covers five different categories, namely, **Nvdla-small, RISCY, RISCY-FPU, Zero-riscy,** and **Vortex-small**.
- We use **PyCharm** to implement our work in **Python**. All experiments are conducted on an Intel Xeon dual-CPU server with a main frequency of 2.6GHz. The server is equipped with 256GB of RAM, and the graphics card model is **NVIDIA 3070**.

Experiment Setup

We use the correlation coefficient (**CC**), the mean absolute error (**MAE**), and the mean absolute percentage error (**MAPE**) as measures of accuracy.

- The magnitude of the CC indicates the strength of the **linear relationship** between variables.
- The MAE is a metric used to measure the **prediction error** of the model.
- The value of MAPE is expressed as a **percentage and represents the magnitude of the mean relative error**.

$$CC = \frac{\sum_{i=1}^{n} [y_i - \text{mean}(y)] [\hat{y}_i - \text{mean}(\hat{y})]}{\sqrt{\sum_{i=1}^{n} [y_i - \text{mean}(y)]^2 \sum_{i=1}^{n} [\hat{y}_i - \text{mean}(\hat{y})]^2}} \quad (1)$$

$$MAE = \frac{1}{n} \sum_{1}^{n} |\hat{y}i - yi| \quad (2)$$

$$MAPE = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\hat{y}i - yi}{\hat{y}i} \right| \times 100 \quad (3)$$

Experiment Results

Table 2: Unet and UnetPro test results for each metric under the four category datasets.

Model	Zero-riscy		RISCY		RISCY-FPU		Nvdla-small	
	MAE(V)	CC	MAE(V)	CC	MAE(V)	CC	MAE(V)	CC
Unet	0.0016	0.7953	0.0013	0.7369	0.0014	0.7459	0.0028	0.7219
UnetPro	0.0017	0.8786	0.0011	0.8830	0.0012	0.8345	0.0021	0.8178

- UnetPro outperforms Unet with lower MAE and higher CC. The correlation coefficient is higher than **85%**.
- UnetPro performs better than Unet across all categories, indicating its wider adaptability and performance advantages.
- For the UnetPro model, performance is better on RISCY and RISCY-FPU processors, with higher MAE and CC values. The mutual adaptation between the characteristics of the RISCY and RISCY-FPU data classes and the UnetPro model may be the main reason for the better performance on these processors.

Experiment Results

We take an additional **10 cases** from the data and make further comparison tests between the Unet model and the UnetPro model using the MAPE metric on them, as shown in Figure 9.

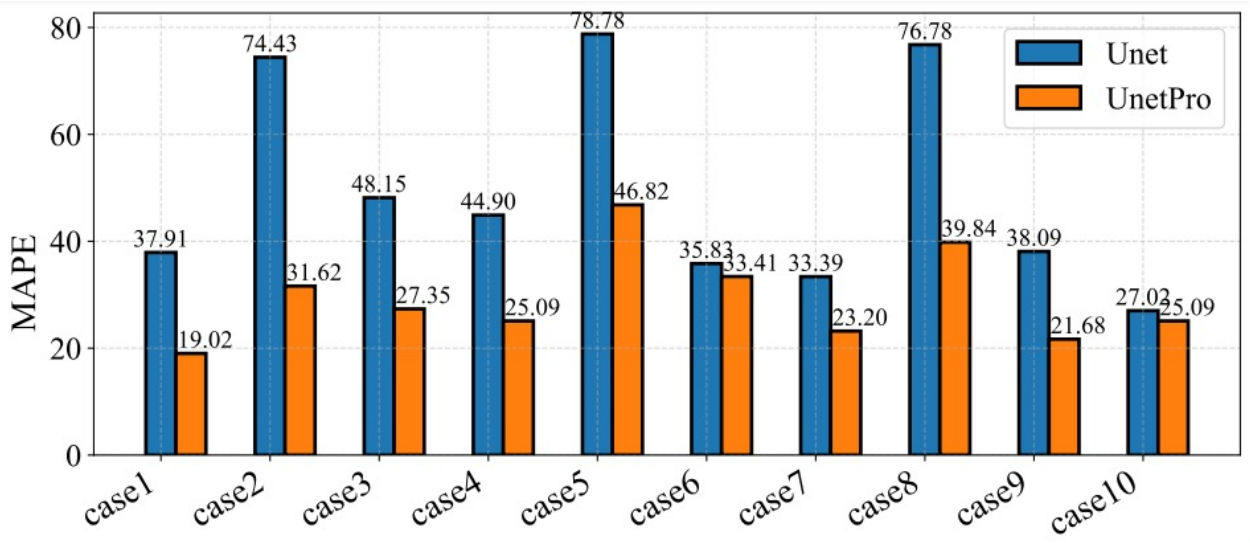


Figure 9: Data segmentation diagram.

- The average MAPE error of the UnetPro model is about **20%** lower than that of the Unet model.
- This indicates that UnetPro is significantly **smaller than Unet in terms of prediction error**, which is smaller and the model is **more accurate** in its predictions.

Conclusions

- We introduce **UnetPro** as an IR drop prediction model.
- The prediction accuracy of the model are significantly improved by combining structures such as **attention**, **skip connection**, and **multi-scale convolution**.
- **Data enhancement** further improves model performance.
- Experiments demonstrate that UnetPro outperforms Unet with lower MAE and higher CC and the correlation coefficient is higher than **85%**.

Future work

1) Can segmentation accuracy and gradient transfer still be optimized?

In future work, we will design **dense skip connections** (shown in blue), which enable hopping paths between the encoder and decoder to generate **higher resolution feature maps**.

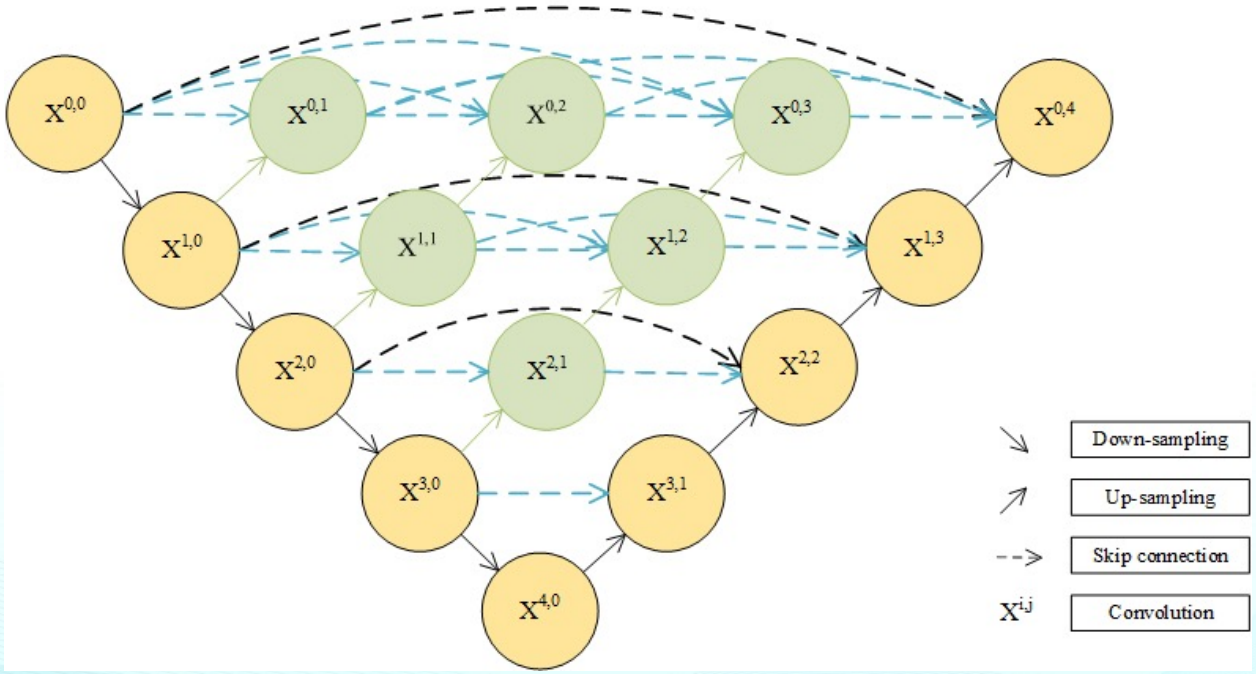


Figure 10: Dense connection diagram.

Future work

2) Is it possible to speed up UnetPro's inference?

In future work, we will design a **pruning scheme** that adds deep supervision **to achieve a balance between speed and performance**.

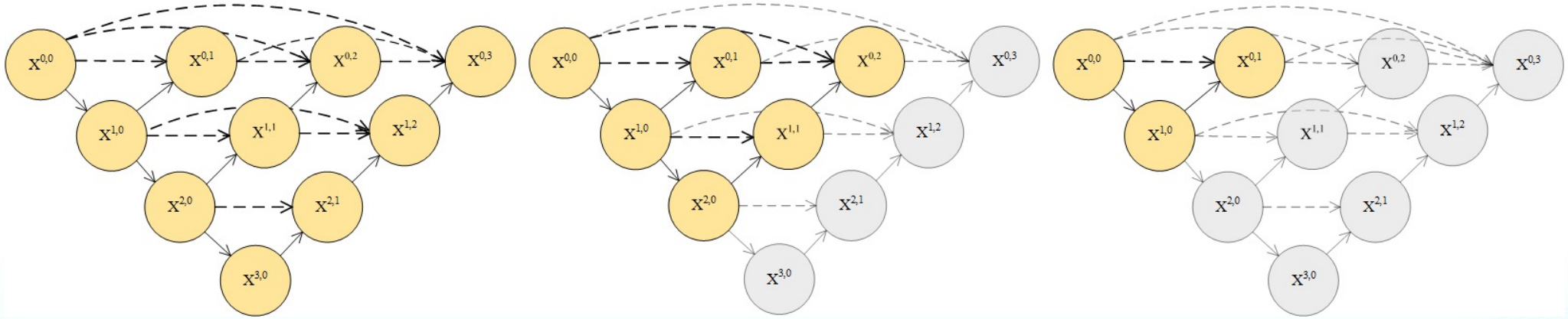


Figure 11: Deep supervision diagram.

THANKS!

If you have any questions, please feel free to contact us!

Email: zhengfei.qi@student.cup.edu.cn