

Применение методов компьютерного зрения в задачах вычислительной литографии

Подготовил: Московцев Андрей

Литографический процесс

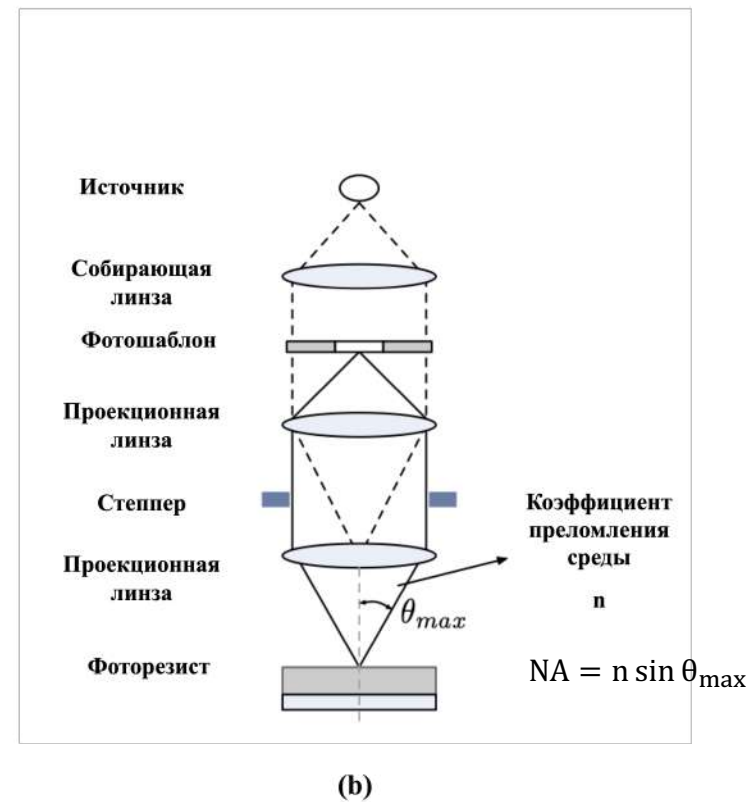
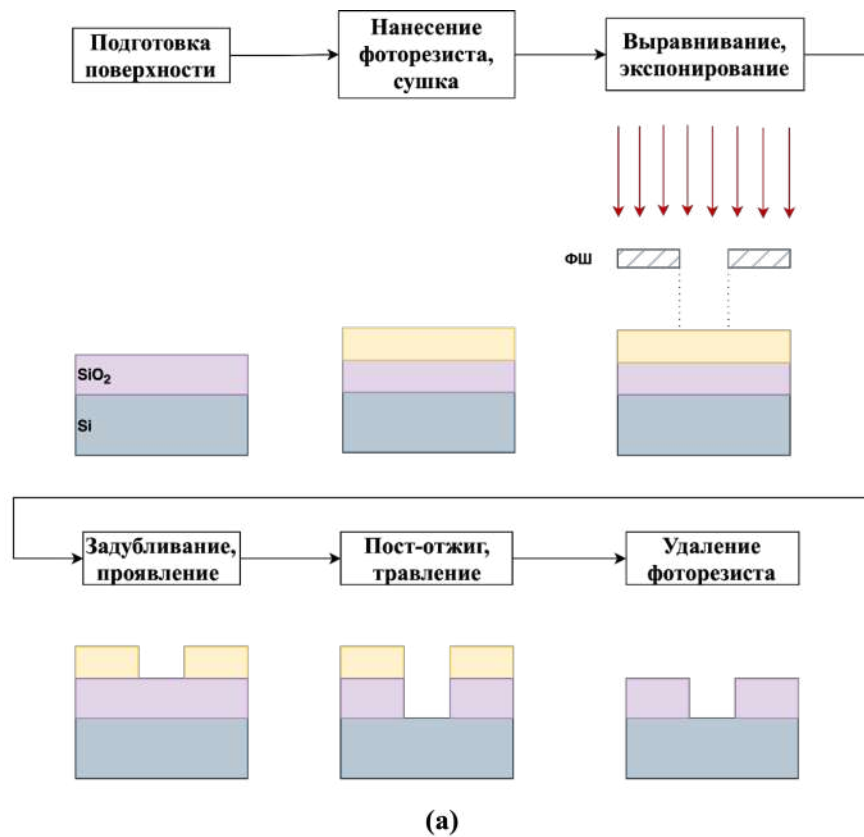


Рис. 1: (a) Технологические этапы процесса литографии;
(b) схема проекционной установки

Ограничения

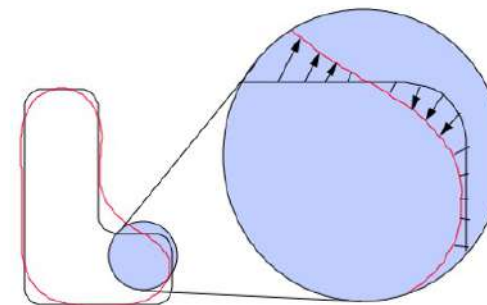
- При уменьшении характерных размеров элементов ниже половины длины волны света начинают сильно проявляться дифракционные эффекты, вызывая размытие и искажение печатаемого изображения (например, скругление углов на краях линий);
- В результате получаемая топология на пластине может значительно отличаться от исходного дизайна проектирования, что может привести к снижению выхода годных и производственным дефектам;
- При современных проектных нормах элементы на ФШ располагаются более плотно друг к другу. Эта близость увеличивает интерференцию между соседними элементами во время экспонирования, вызывая нежелательное взаимодействие;
- Оптические эффекты близости могут привести к нежелательному слиянию или разрыву линий, а также к изменению ширины элементов топологии. В результате на пластине образуются элементы неправильного размера или формы

$$CD = k_1 \frac{\lambda}{NA} = k_1 \frac{\lambda}{n \sin \Theta}$$

$$k_{1min}^{theory} = 0.25$$

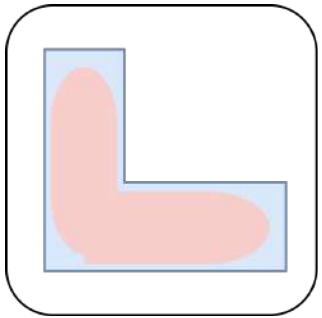
$$k_{1min}^{practice} = 0.3 - 0.6$$

$$\lambda_{DUV} = 193\text{нм}$$

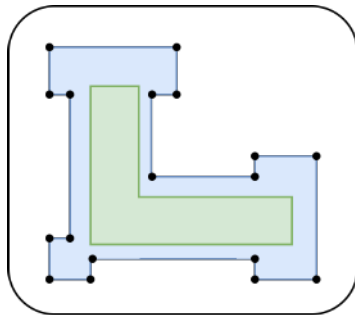


ML-методы для задач вычислительной литографии

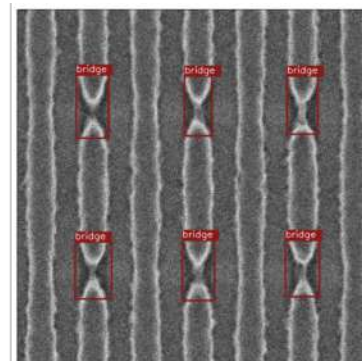
Литографическое
моделирование



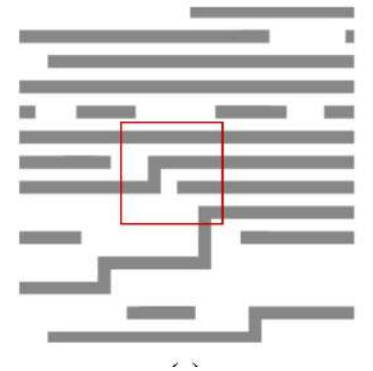
Коррекция эффектов
оптической близости



Анализ SEM-изображений



Поиск “горячих точек”



Литографическое моделирование

Идея подхода для литографического моделирования

[1] Воздушное изображение I может быть аппроксимировано в виде линейной комбинации свёрток маски M с оптическими ядрами H в соответствии моделью дифракции Хопкинса (оптическая модель):

$$I(x, y) = H(M) = \sum_{k=1}^K \mu_k |h_k(x, y) \otimes M(x, y)|^2$$

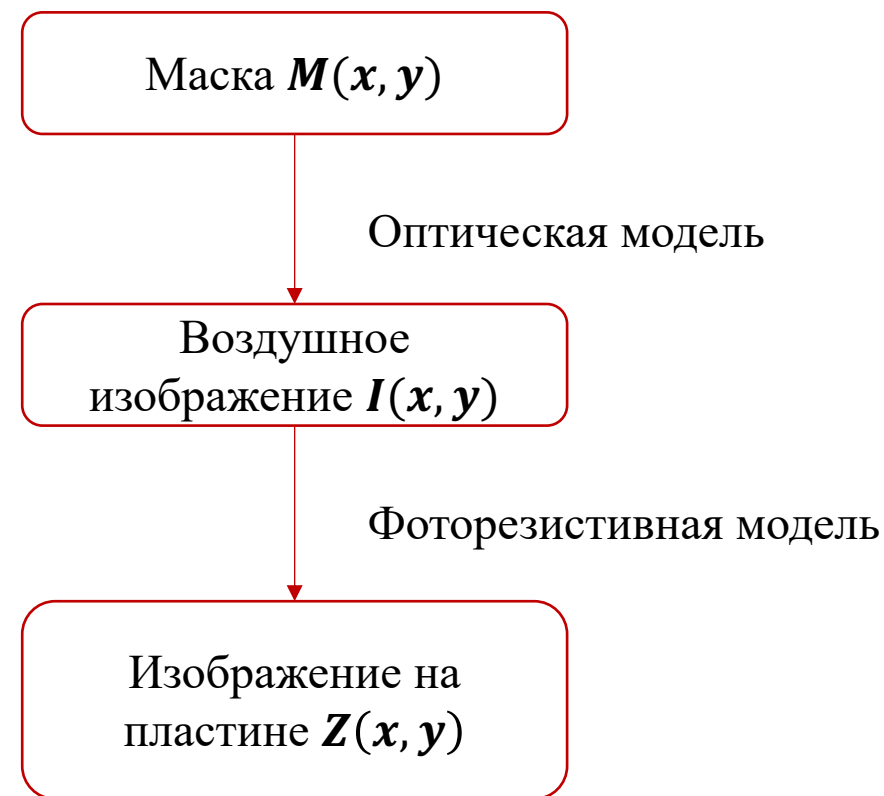
Вес μ_k

k -ое ядро Хопкинса h_k

Маска M

Полученное воздушное изображение (по сути распределение интенсивности на пластине) подаётся на вход фоторезистивной модели, преобразующей его в изображение на пластине Z . Порог I_{th} определяет уровень экспозиции:

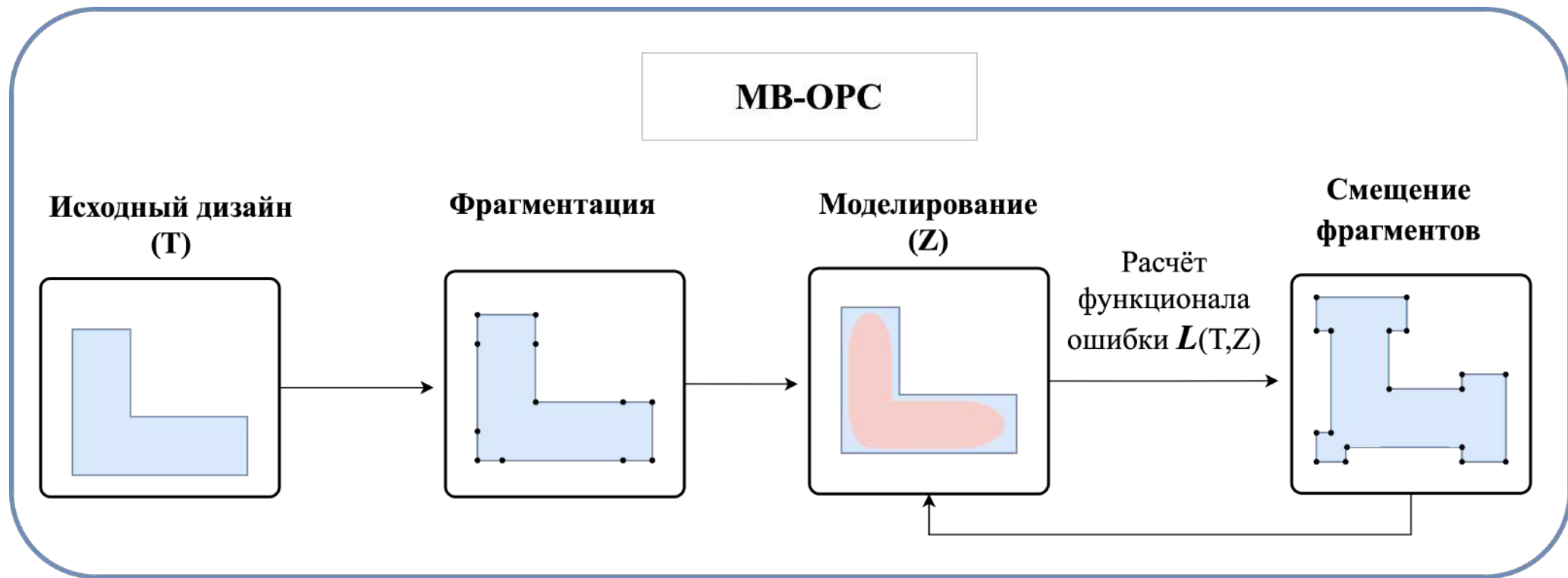
$$Z(x, y) = \sigma_Z(I(x, y)) = \frac{1}{1 + e^{(-\alpha(I(x, y) - I_{th}))}}$$



1. Zheng S., Yu B., Wong M. OpenILT: An open source inverse lithography technique framework //2023 IEEE 15th International Conference on ASIC (ASICON). – IEEE, 2023. – С. 1-4.

Коррекция эффектов оптической близости

Коррекция эффектов оптической близости на основе моделирования (МВОРС)



Идея GAN-подхода для OPC(1/2)

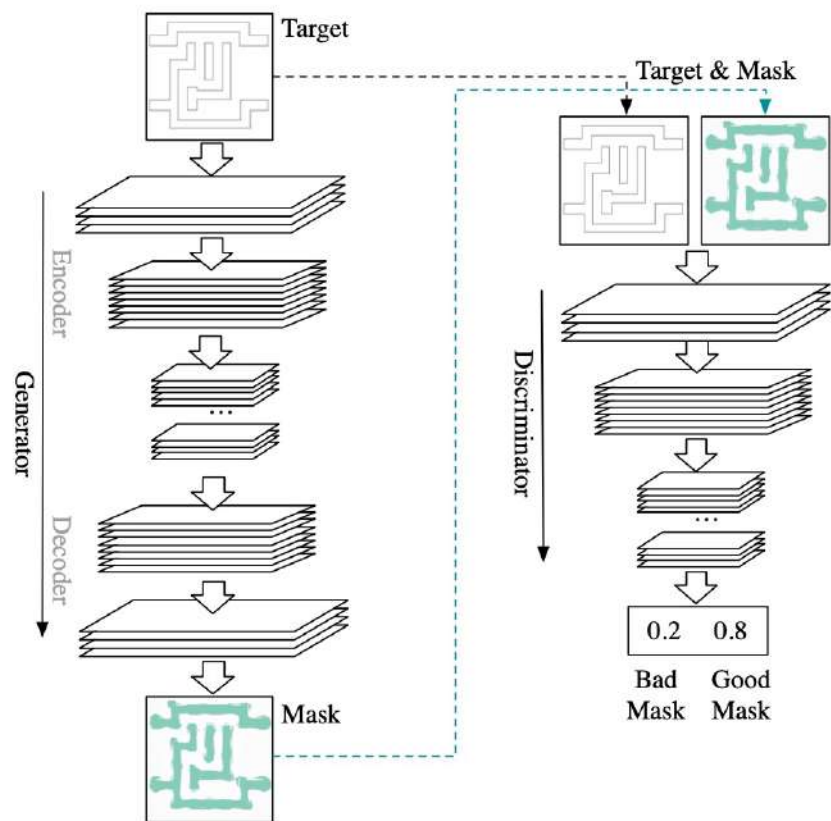


Рис.2[1]: применение GAN для OPC

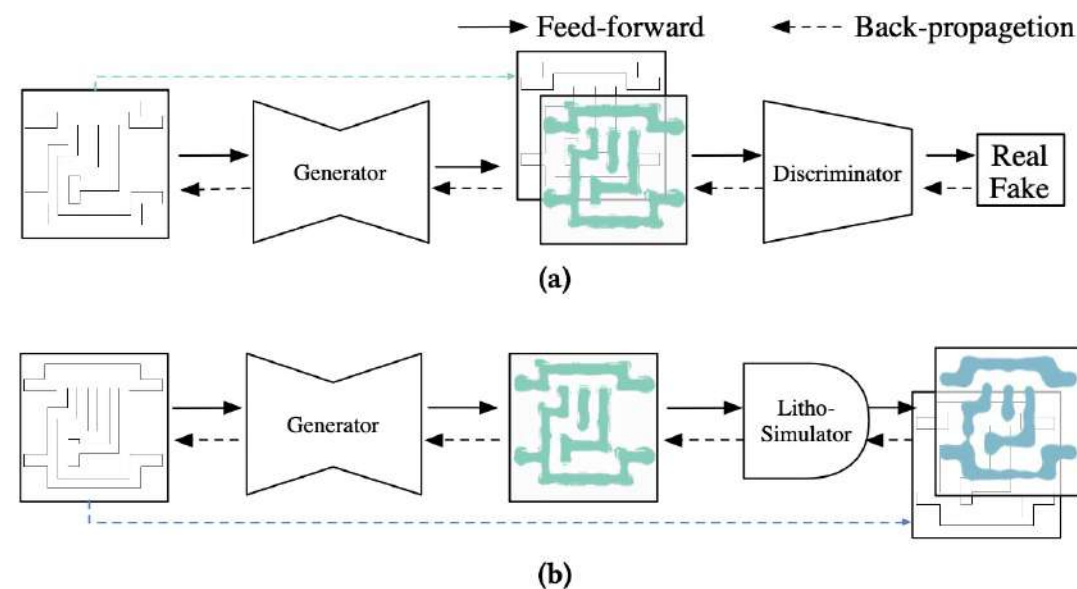


Рис.3[1]: (a)обучение GAN (генератор + дискриминатор);
(b) предварительное обучение генератора с использованием литографического моделирования

1. H. Yang, S. Li, Y. Ma, B. Yu and E. F. Y. Young - "GAN-OPC: Mask Optimization with Lithography-guided Generative Adversarial Nets,"2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, USA, 2018, pp. 1-6, doi: 10.1109/DAC.2018.8465816.

Идея GAN-подхода для OPC(2/2)

Генератор и дискриминатор обучаются совместно, играя в т.н. «минимакс» игру:

$$\min_{\theta_g} \max_{\theta_d} \left[\underbrace{E_{x,y} \log D_{\theta_d}(x, y)}_{\substack{\text{Выход} \\ \text{дискриминатора для} \\ \text{реальных данных}}} + \underbrace{E_x \log(1 - D_{\theta_d}(x, G_{\theta_g}(x)))}_{\substack{\text{Выход} \\ \text{дискриминатора для} \\ \text{сгенерированных} \\ \text{данных}}} \right] + \underbrace{E_{x,y} \|y - G(x)\|_1}_{\substack{\text{L1-функция потерь при условной} \\ \text{генерации}}} + \underbrace{E_x \|x - \text{LithoModel}(G(x))\|_1}_{\substack{\text{Литографическое моделирование} \\ \text{над генерируемой коррекцией} \\ G(x, z)}}$$

Дискриминатор D_{θ_d} стремится максимизировать целевую функцию, чтобы $D_{\theta_d}(x, y)$ было близко к 1 (**истинный объект**), а $D_{\theta_d}(x, G_{\theta_g}(x))$ было близко к 0 (**сгенерированный объект**):

$$\max_{\theta_d} \left[E_{x,y} \log D_{\theta_d}(x, y) + E_x \log(1 - D_{\theta_d}(x, G_{\theta_g}(x))) \right]$$

Генератор G_{θ_g} стремится минимизировать целевую функцию таким образом, чтобы $D_{\theta_d}(x, G_{\theta_g}(x))$ был близок к 1 (дискриминатор обманут, полагая, что сгенерированный объект является реальным):

$$\min_{\theta_g} \left[E_{x,z} \log(1 - D_{\theta_d}(x, G_{\theta_g}(x))) \right]$$

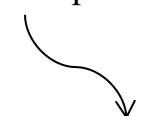
На практике вместо минимизации целевой функции у генератора $\log(1 - D(x, G(x)))$ максимизируют $\log(D(x, G(x)))$:

$$\max_{\theta_g} \left[E_x \log(D_{\theta_d}(x, G_{\theta_g}(x))) \right]$$

Метрики

- **PixelAccuracy** (попиксельная точность) – метрика, представляющая собой отношение числа верно предсказанных значений пикселей у сгенерированной маски к общему числу пикселей на изображении.
- **IoU** (Intersection Over Union) – метрика, представляющая собой отношение площади пересечения предсказанной и целевой коррекции к площади их объединения.
- **t_{gen}** – среднее время генерации изображения откорректированной топологической структуры, с

длина x ширина


$$\text{PixelAccuracy} = \frac{\sum_{i=0}^{N \times N} \mathbb{I}[\text{target}_i = \text{prediction}_i]}{N \times N}$$

$$\text{IoU} = \frac{\sum_{i=0}^{N \times N} \text{target}_i * \text{predicition}_i}{[\sum_{i=0}^{N \times N} \text{target}_i + \text{predicition}_i] - [\sum_{i=0}^{N \times N} \text{target}_i * \text{predicition}_i]}$$

Функции потерь

- **Binary Cross Entropy** (бинарная кросс-энтропия) – минимизация расхождения между предсказанием и истинным значением на основе вероятностного распределения

$$L_{\text{BCE}} = -\frac{1}{B} \frac{1}{N \times N} \sum_{i=1}^B \sum_{j=1}^{N \times N} y_{ij} \log p_{ij} + (1 - y_{ij}) \log(1 - p_{ij})$$

длина x ширина

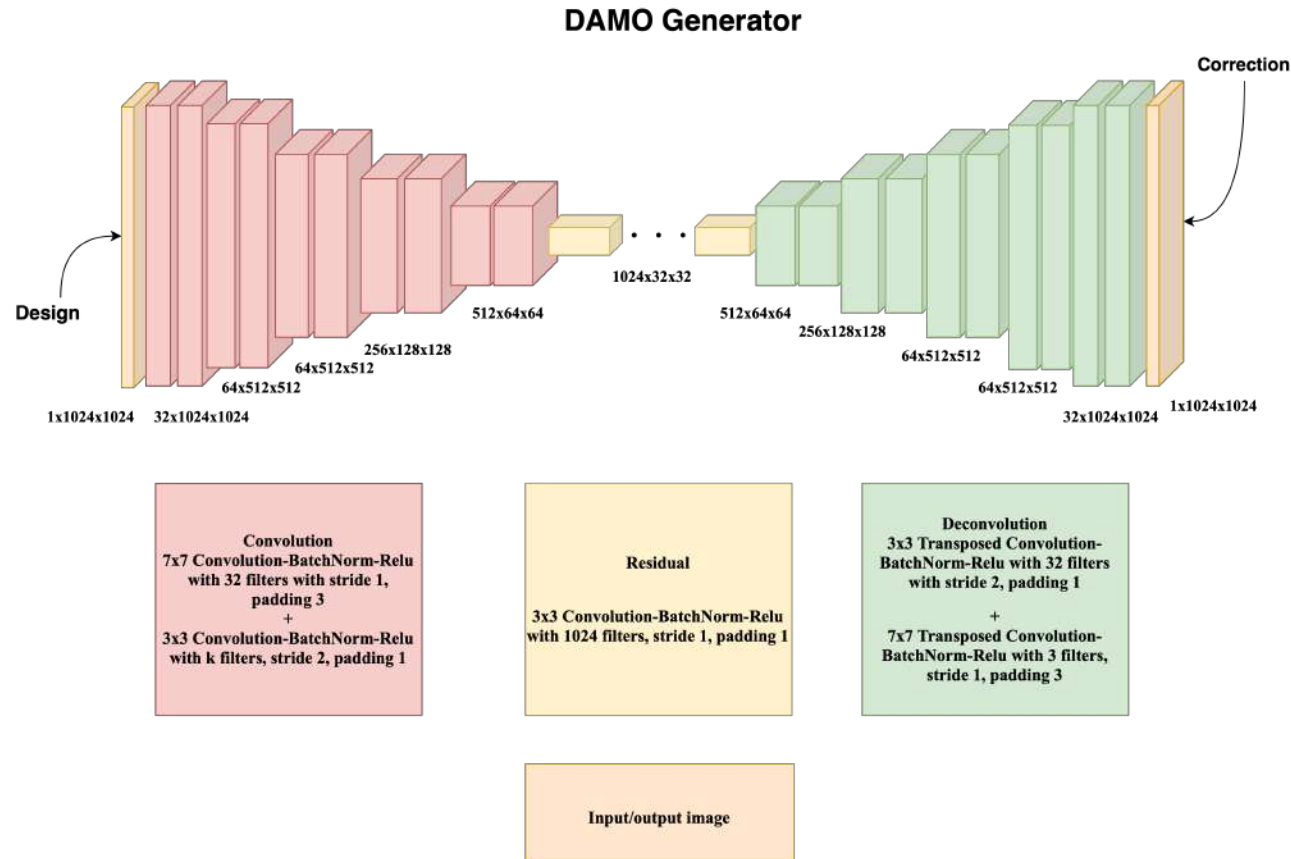
батч

вероятность того, что пиксель является полигоном

- **IoU Loss** – максимизация площади пересечения между истинным объектом и предсказанием

$$L_{\text{IoU}} = 1 - \text{IoU} = 1 - \frac{\sum_{i=1}^B \sum_{j=0}^{N \times N} \text{target}_{ij} * \text{predicton}_{ij}}{\left[\sum_{i=1}^B \sum_{j=0}^{N \times N} \text{target}_{ij} + \text{predicton}_{ij} \right] - \left[\sum_{i=1}^B \sum_{j=0}^{N \times N} \text{target}_{ij} * \text{predicton}_{ij} \right]}$$

Иллюстрация подхода



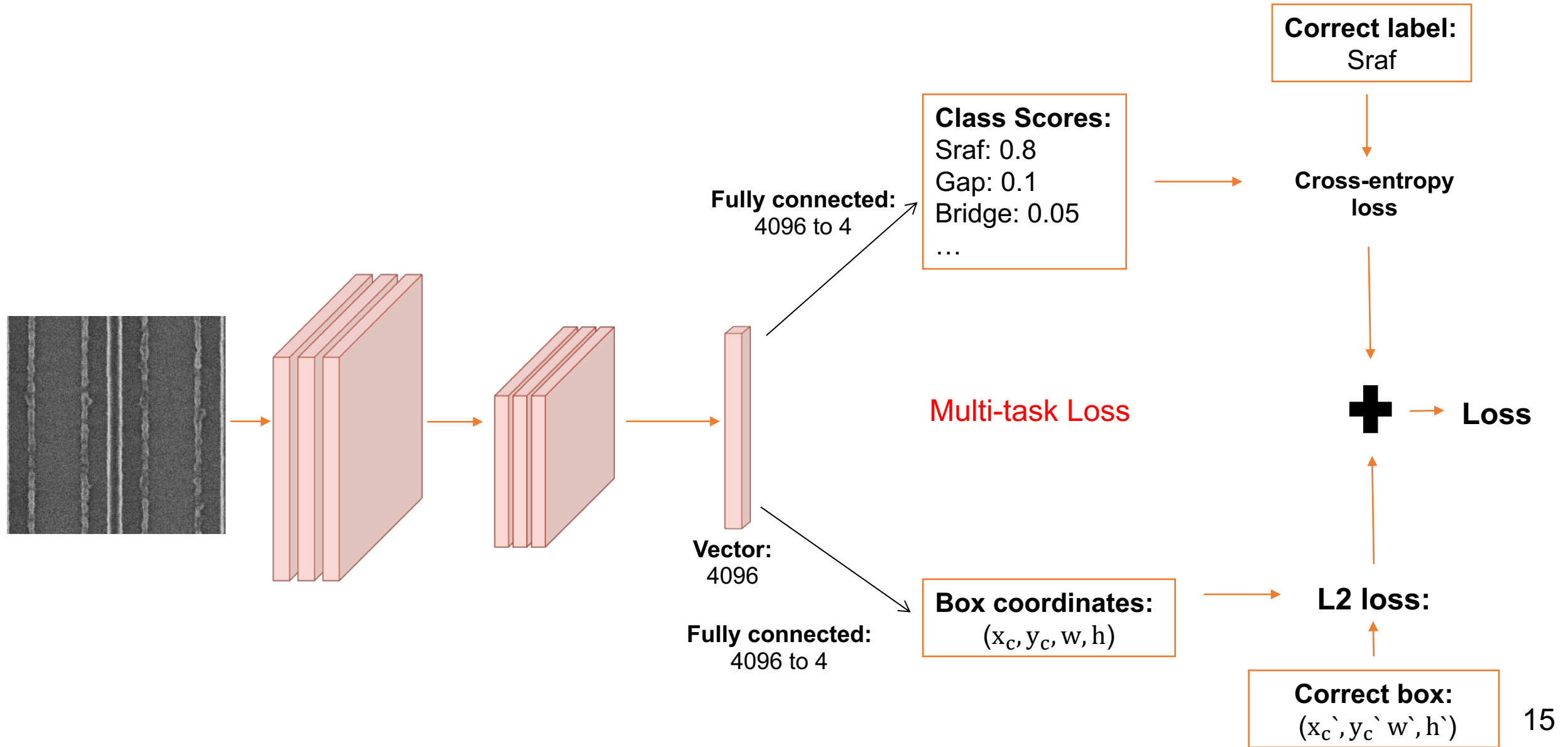
$$\min_w L(y, \text{model}(x))$$

Рис. 2: Архитектура генератора DAMO[1], использованная в качестве baseline

1. Zheng, S., Yang, H., Zhu, B., Yu, B., & Wong, M. (2023). Lithobench: Benchmarking ai computational lithography for semiconductor manufacturing. *Advances in Neural Information Processing Systems*, 36, 30243-30254.

Анализ SEM-изображений

Задача детекции в общем виде



Выход модели

Размер выходного тензора: $V \times (5 + K)$

K – число классов,

V – число ограничивающих рамок

$$5 = |x_c, y_c, w, h, p_c|$$

x_c, y_c – нормированные координаты центров ограничивающих рамок

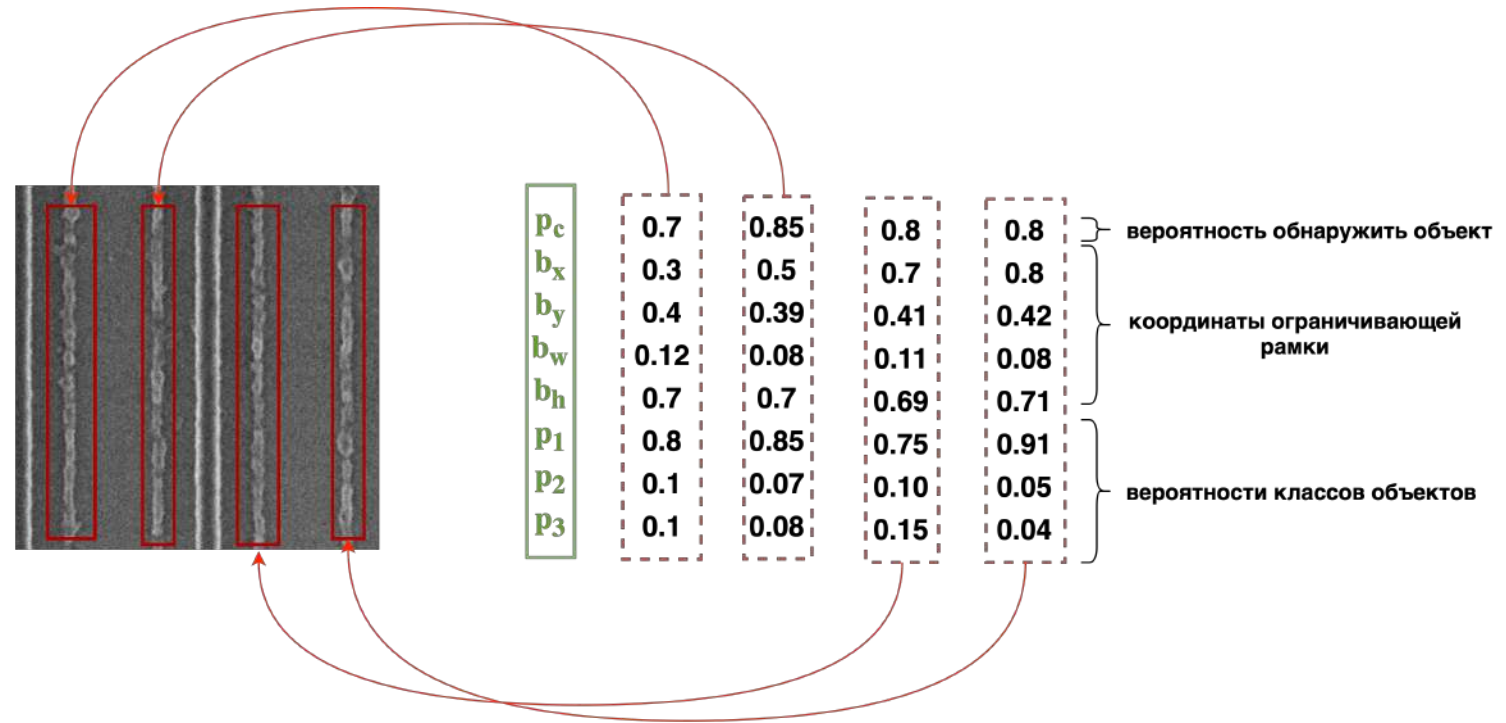
w, h – ширина и высота ограничивающей рамки

p_c – вероятность обнаружить объект внутри рамки

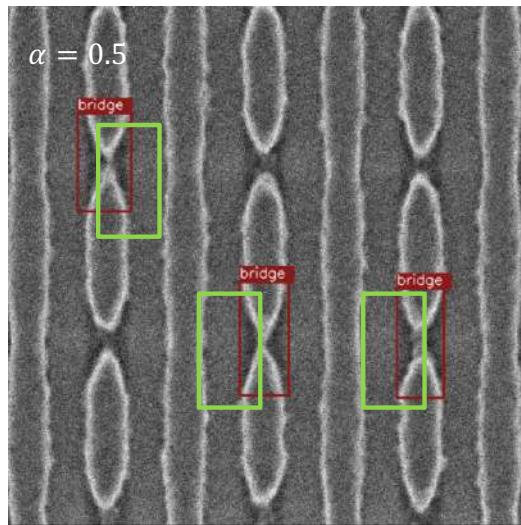
P_1 – вероятность обнаружить объект класса 1

P_2 – вероятность обнаружить объекта класса 2

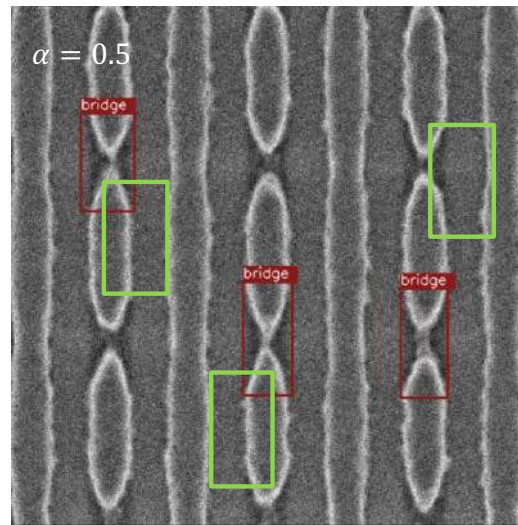
P_3 – вероятность обнаружить объекта класса 3



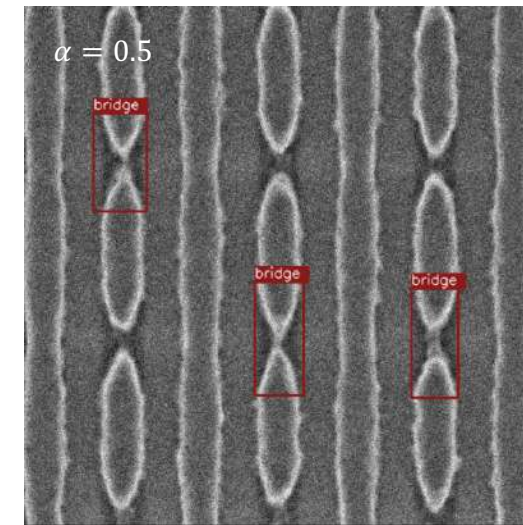
Intersection over Union (IoU)



IoU = 0.80
True Positive(TP)
(a)



IoU = 0.22
False Positive(FP)
(b)



IoU = 0
False Negative(FN)
(c)

$$J(B_p, B_{gt}) = IoU = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

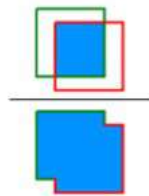
B_{gt} - истинная рамка

B_p - предсказанная рамка

α - порог

$IoU > \alpha \rightarrow$ верно

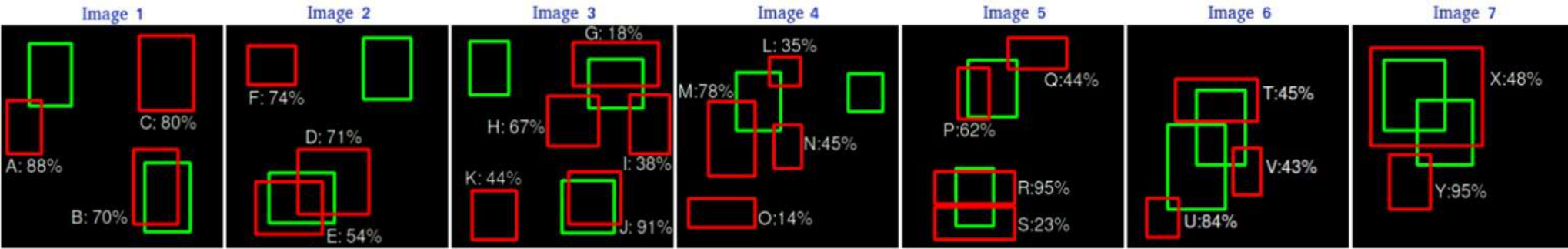
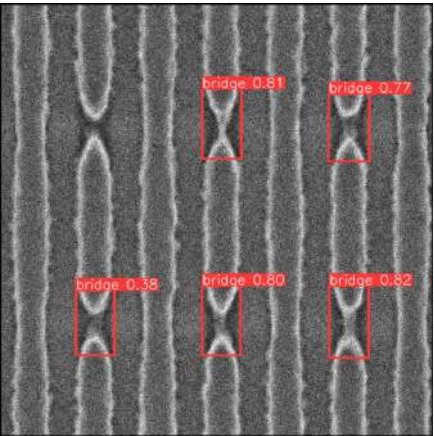
$IoU < \alpha \rightarrow$ неверно



- **True positive(TP):** верное предсказание истинного расположения рамки объекта;
- **False positive(FP):** детектирование несуществующего объекта или неверное детектирование существующего объекта;
- **False Negative(FN):** алгоритм не смог обнаружить истинное расположение рамки объекта

В контексте задач детекции **True Negative(TN)** не используется, поскольку существует бесконечное количество ограничивающих рамок, предсказывающих фон изображения, которые не нужно детектировать.

Оценка точности алгоритмов детекции



$$AP_{all} = \sum_n (R_{n+1} - R_n) P_{interp}(R_{n+1})$$

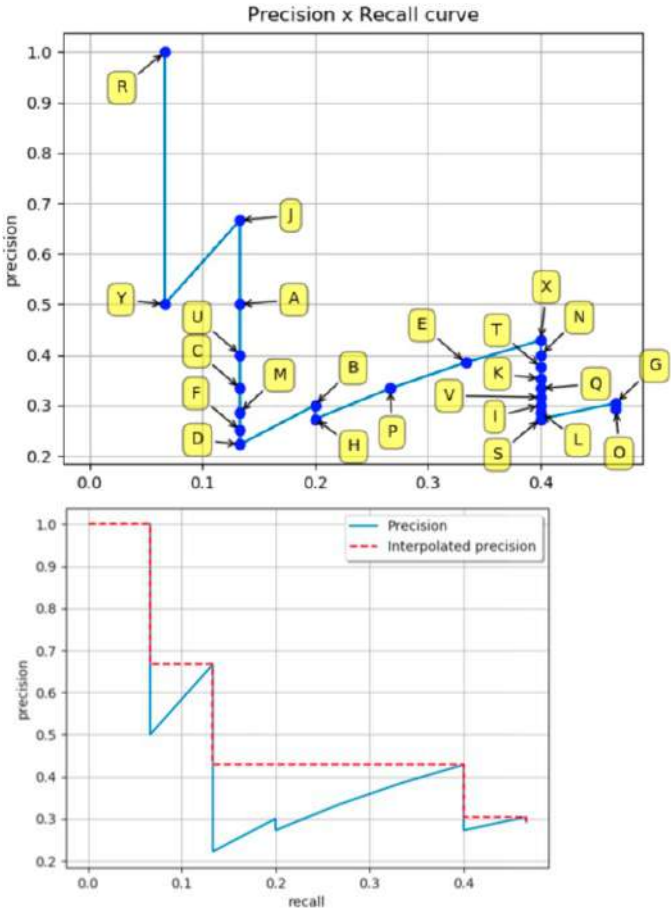
$$P_{interp}(R_{n+1}) = \max_{\tilde{R}: \tilde{R} \geq R_{n+1}} P(\tilde{R})$$

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths}$$

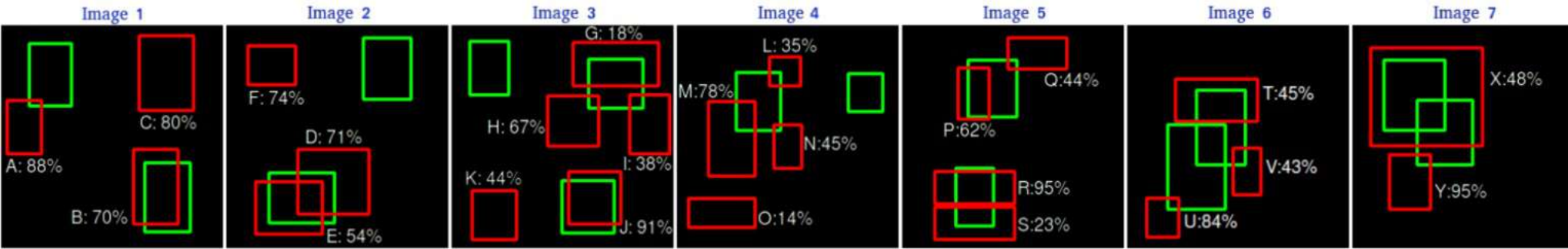
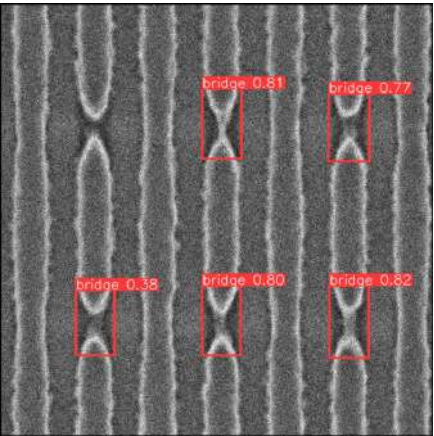
$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

$AP_i \rightarrow i$ – ый класс



Detection	confidence	TP	FP	Acc TP	Acc FP	precision	recall
R	95%	1	0	1	0	1	0.066
Y	95%	0	1	1	1	0.5	0.066
J	91%	1	0	2	1	0.666	0.133
A	88%	0	1	2	2	0.5	0.133
U	84%	0	1	2	3	0.4	0.133
C	80%	0	1	2	4	0.333	0.133
M	78%	0	1	2	5	0.286	0.133
F	74%	0	1	2	6	0.25	0.133
D	71%	0	1	2	7	0.222	0.133
B	70%	1	0	3	7	0.3	0.2
H	67%	0	1	3	8	0.273	0.2
P	62%	1	0	4	8	0.333	0.266
E	54%	1	0	5	8	0.385	0.33
X	48%	1	0	6	8	0.423	0.4
N	45%	0	1	6	9	0.4	0.4
T	45%	0	1	6	10	0.375	0.4
K	44%	0	1	6	11	0.353	0.4
Q	44%	0	1	6	12	0.333	0.4
V	43%	0	1	6	13	0.316	0.4
I	38%	0	1	6	14	0.3	0.4
L	35%	0	1	6	15	0.286	0.4
S	23%	0	1	6	16	0.273	0.4
G	18%	1	0	7	16	0.304	0.467
O	14%	0	1	7	17	0.292	0.467

Оценка точности алгоритмов детекции



$$AP_{all} = \sum_n (R_{n+1} - R_n) P_{interp}(R_{n+1})$$

$$P_{interp}(R_{n+1}) = \max_{\tilde{R}: \tilde{R} \geq R_{n+1}} P(\tilde{R})$$

$$P = \frac{TP}{TP + FP} = \frac{TP}{all\ detections}$$

$$R = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths}$$

$$mAP = \frac{1}{N} \sum_{i=1}^N AP_i$$

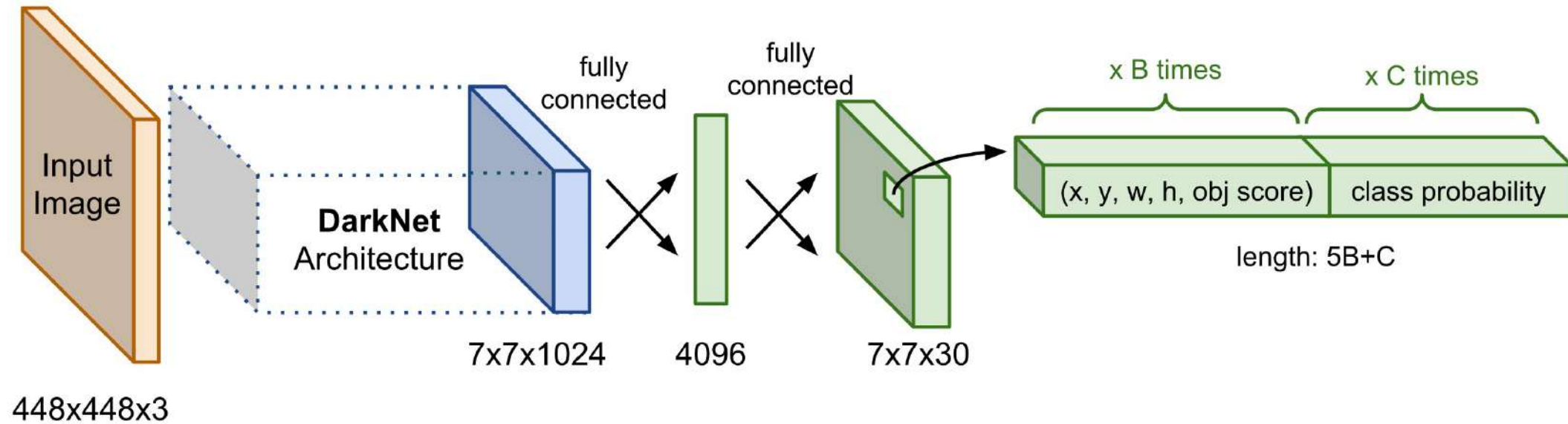
$AP_i \rightarrow i$ – ый класс

Значение mAP@50	Качество детекции
50% - 60%	низкое
60%-70%	умеренное
70%-80%	хорошее
Выше 80%	высокое

Значение mAP@50:95	Качество детекции
20% - 30%	низкое
30%-40%	умеренное
40%-50%	хорошее
Выше 50%	высокое

$$\begin{matrix} AP_{0.5} & - & mAP_{0.5} \\ AP_{0.5:0.95} & - & mAP_{0.5:0.95} \end{matrix}$$

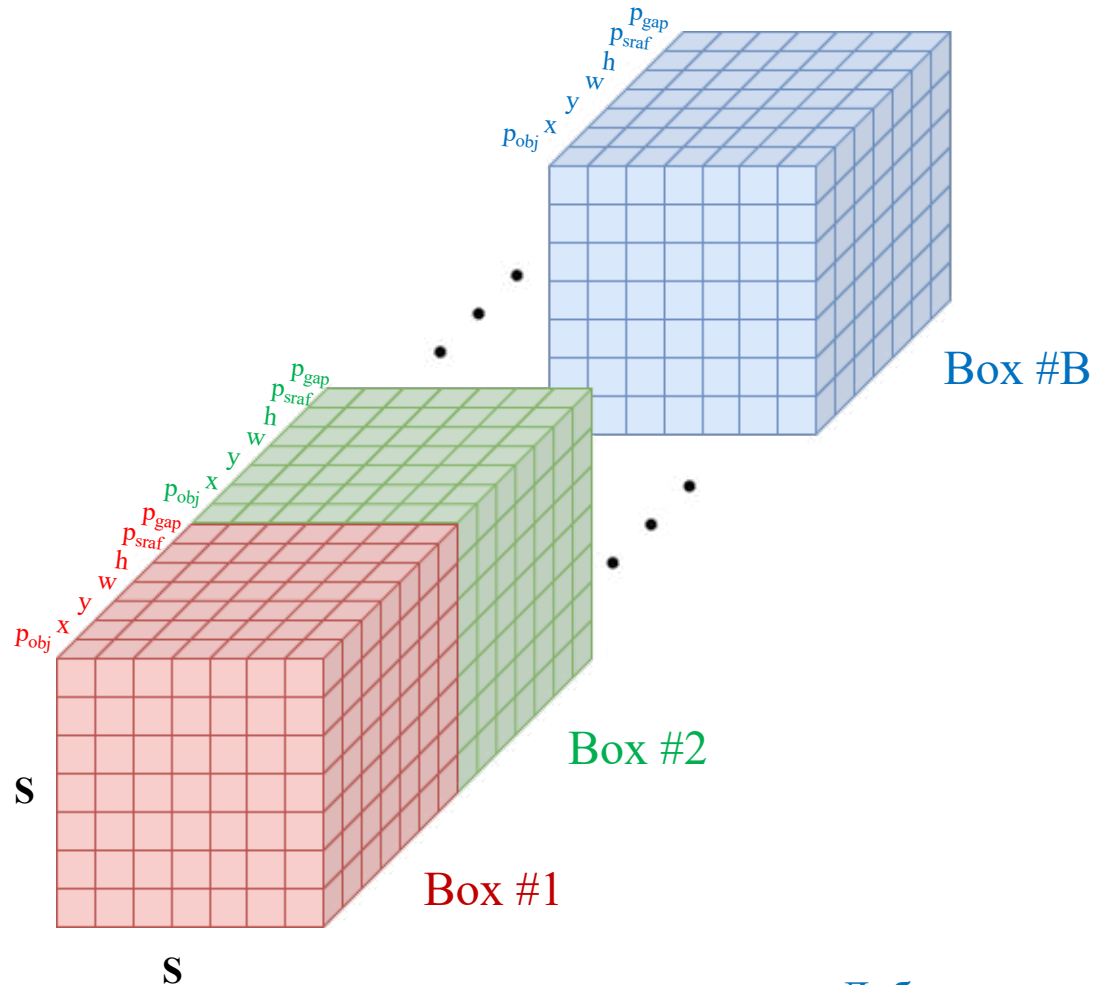
Архитектура 1-го детектора Yolo



[\[paper\]](#)

[\[blog\]](#)

Выход модели



Размер выходного тензора: $S \times S \times (5 + K) \times B$

$S \times S$ – размер сетки для исходного изображения

K – число классов,

B – число ограничивающих рамок на каждую ячейку сетки

$5 = |x, y, w, h, p_{obj}|$

x, y – смещение от начала координат $(0,0)$ ячейки до центра объекта внутри ограничивающей рамки

w, h – ширина и высота ограничивающей рамки

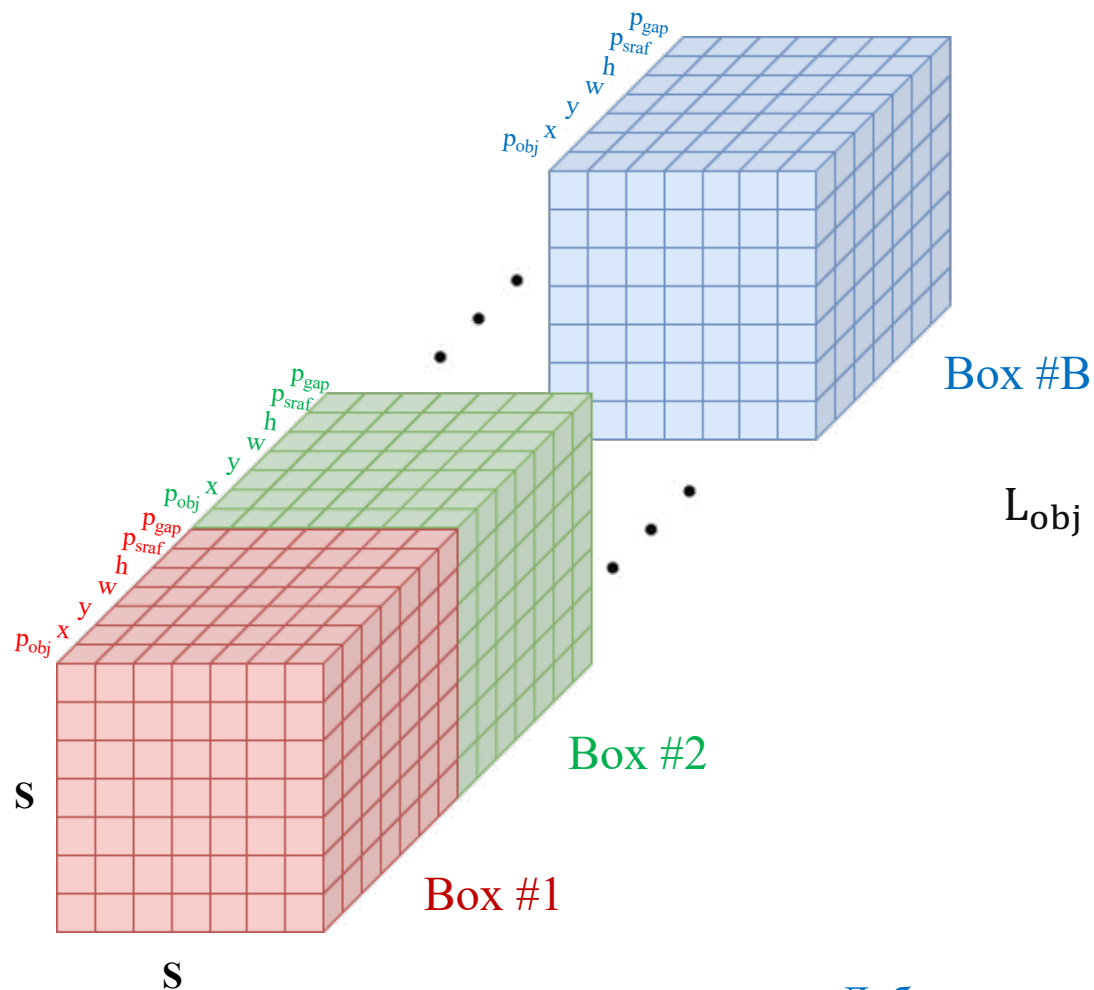
P_{obj} – вероятность обнаружить объект внутри рамки

P_{sraf} – вероятность обнаружить объект класса "sraf"

P_{gap} – вероятность обнаружить объекта класса "gap"

Дублирующиеся предсказания удаляются алгоритмом NMS
(non-maximum suppression – подавление немаксимумов)

Функция потерь



$$L = \lambda_{cls} L_{cls} + \lambda_{obj} L_{obj} + \lambda_{reg} L_{loc}$$

$$L_{cls} = \alpha_{class} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{obj} \sum_{k \in K} \hat{p}_i(k) \log(p_i(k))$$

$$L_{obj} = \alpha_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{I}_{ij}^{noobj} (c_i - \hat{c}_i)^2 + \alpha_{obj} \sum_{i=0}^{S^2} \sum_{j=0}^N \mathbb{I}_{ij}^{obj} (c_i - \hat{c}_i)^2$$

$$L_{loc} = 1 - GIoU = 1 - IoU + \frac{C \setminus (B_{pr} \cap B_{gt})}{C}$$

Дублирующиеся предсказания удаляются алгоритмом NMS
(non-maximum suppression – подавление немаксимумов)

Принцип работы на примере 1-го детектора Yolo

- Разбиваем изображения на $S \times S$ блоков:
- Для каждого блока предсказываем B прямоугольников
- Для каждого прямоугольника:
 - Координаты
 - Вероятность наличия объекта
 - Вероятность каждого класса при условии наличия объекта

NMS

- Модель выдаёт для класса дефект#1 список прямоугольников с различными confidence score
- Выбираем прямоугольник с максимальным confidence score, убираем его из общего списка
- Сортируем оставшиеся прямоугольники в порядке убывания confidence score
- Удаляем каждый последующий прямоугольник, имеющий IoU с первым > 0.5

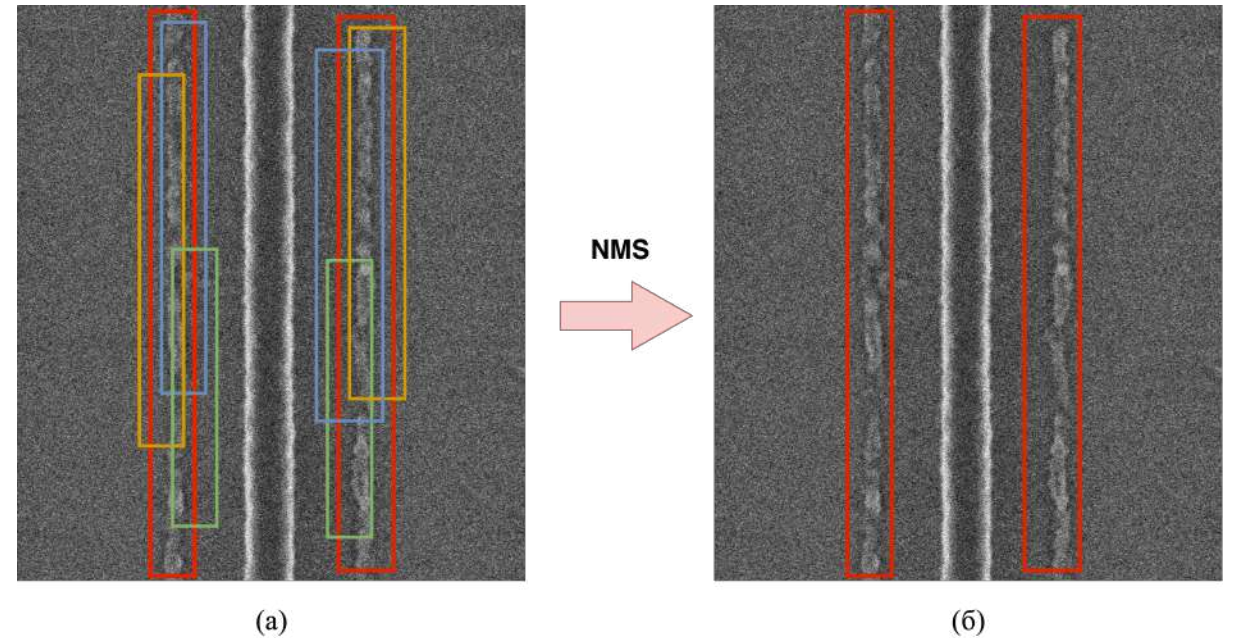


Рис.: Визуализация алгоритма NMS.

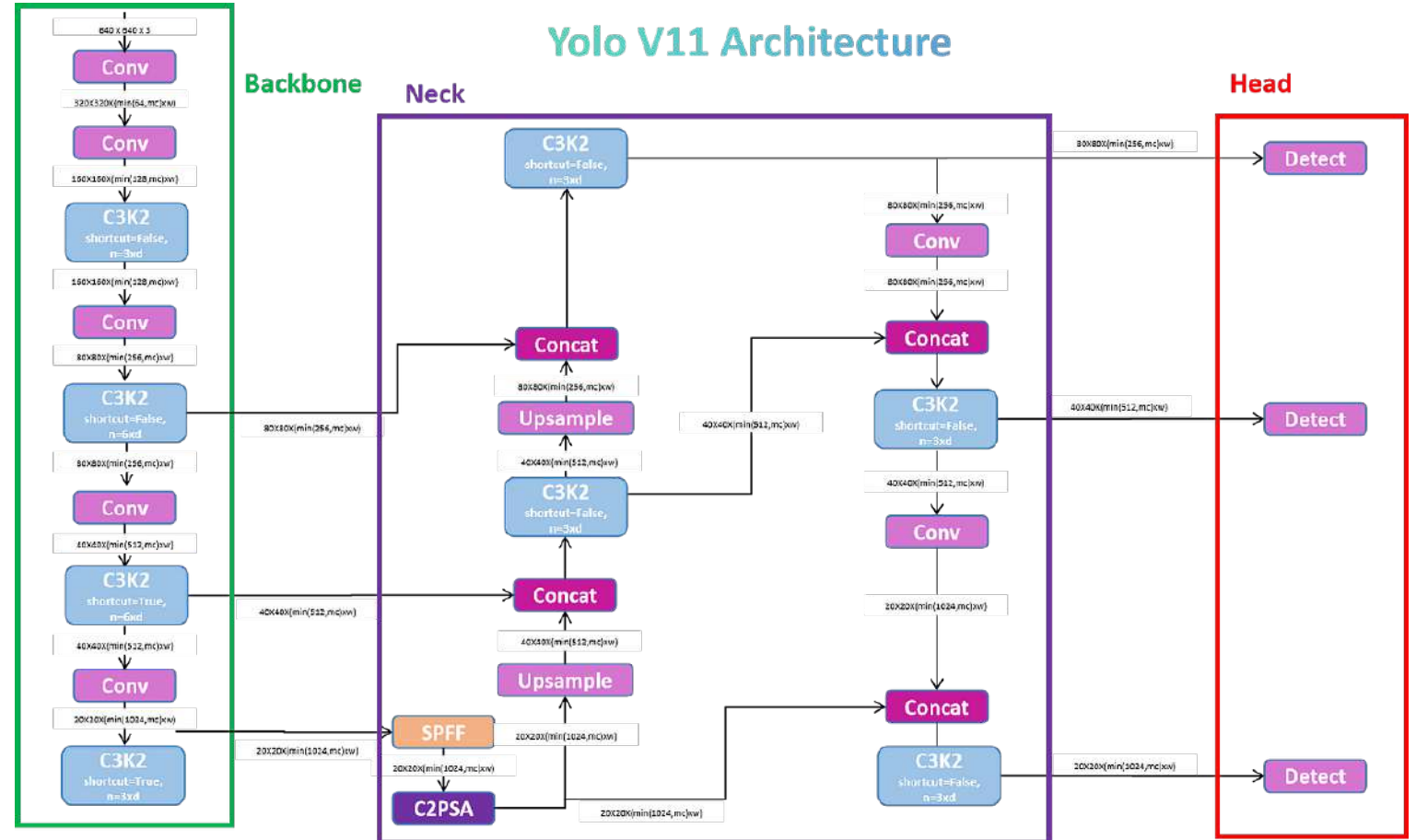
(а) Показан промежуточный результат предсказания модели детекции; (б) Результат постобработки после применения алгоритма NMS

Чем пользуются сейчас?

Backbone: извлечение признаков входного изображения

Neck: агрегация карт признаков разного масштаба (комбинирование карт признаков с различных слоёв backbone)

Head: предсказания для объектов 3-ёх уровней масштаба (малые, большие, средние)



Поиск горячих точек

Идея подхода для поиска горячих точек(1/3)

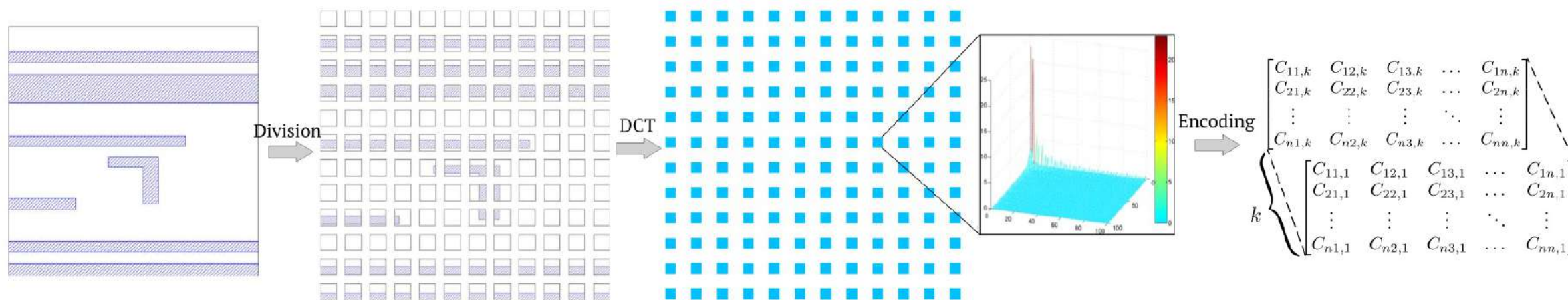


Рис. 10[4]: Пример генерации признаков из топологии ($n = 12$). Область топологии (1200x1200 нм) делится на 12x12 блоков. Каждый блок переводится в изображение размером 100x100px, представляющий область 100x100 нм² на исходной топологии. Данные для обучения получаются путём применения дискретного косинусного преобразования ко всем блокам. От каждого блока берутся только первые k-коэффициентов

Идея подхода для поиска горячих точек(2/3)

Шаг 1: разбить топологию на $n \times n$ блоков с целью генерации признаков для каждого региона

Шаг 2: перевести каждый регион $I_{i,j}$, ($i, j = 0, 1, \dots, n - 1$) в пространство частот

$$D_{i,j}(m, n) = \sum_{x=0}^B \sum_{y=0}^B I_{i,j}(x, y) \cos \left[\frac{\pi}{B} \left(x + \frac{1}{2} \right) m \right] \times \cos \left[\frac{\pi}{B} \left(y + \frac{1}{2} \right) n \right]$$

где (x, y) – индексы положения региона на топологии, (m, n) – индексы в пространстве частот, B – размер региона

Шаг 3: преобразовать множество $D_{i,j}$ в векторы, расположив в порядке убывания низкочастотных компонент:

$$C_{i,j}^* = [D_{i,j}(0,0), D_{i,j}(0,1), D_{i,j}(1,0), \dots, D_{i,j}(B, B)]$$

Шаг 4: взять первые $k \ll B \times B$ элементов от каждого $C_{i,j}^*$. Собрать их в матрицу признаков $F \in \mathbb{R}^{n \times n \times k}$, сохранив их пространственное расположение

$$\begin{bmatrix} c_{11} & \dots & c_{1n} \\ \vdots & \vdots & \vdots \\ c_{n1} & \dots & c_{nn} \end{bmatrix}$$

Идея подхода для поиска горячих точек(3/3)

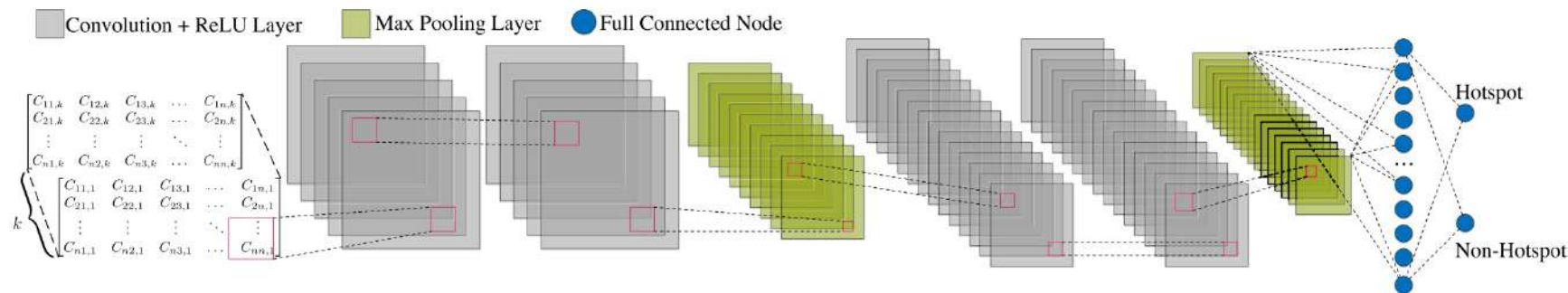


Рис. 11[4]: схема свёрточной нейронной сети для задачи бинарной классификации областей топологии (hotspot – non hotspot)

Layer	Kernel Size	Stride	Output Node #
conv1-1	3	1	$12 \times 12 \times 16$
conv1-2	3	1	$12 \times 12 \times 16$
maxpooling1	2	2	$6 \times 6 \times 16$
conv2-1	3	1	$6 \times 6 \times 32$
conv2-2	3	1	$6 \times 6 \times 32$
maxpooling2	2	2	$3 \times 3 \times 32$
fc1	-	-	250
fc2	-	-	2

Таблица: 1 предлагаемая архитектура свёрточной нейронной сети

4.Yang H. et al. Layout hotspot detection with feature tensor generation and deep biased learning //Proceedings of the 54th Annual Design Automation Conference 2017. – 2017. – С. 1-6.