Multi-GPU системы. Нейронные сети.

- Лектор:
  - Боресков А.В. (ВМиК МГУ)
  - Харламов А.А. (Nvidia)

Multi-GPU

- В CUDA нет встроенной поддержки нескольких GPU

Multi-GPU

- RT API (cuda*) Driver API (cu*)
- Явный выбор устройства:
  - cudaGetDeviceCount()
  - cudaSetDevice
- Работа с разными cuda устройствами должна вестись из разных cpu тредов

Multi-GPU

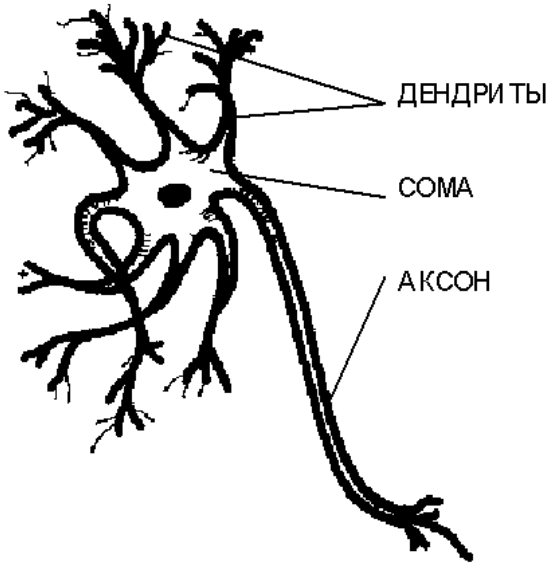
```
int nElem = 1024;
cudaGetDeviceCount(&num_gpus);

if(num_gpus >= 1)
{
    omp_set_num_threads(num_gpus);
#pragma omp parallel
    {
        unsigned int cpu_thread_id = omp_get_thread_num();
        unsigned int num_cpu_threads = omp_get_num_threads();

        cudaSetDevice(cpu_thread_id % num_gpus); // установить device

        dim3 gpu_threads(128);
        dim3 gpu_blocks(nElem / (gpu_threads.x * num_cpu_threads));

        // memory allocation and initialization

        int startIdx = cpu_thread_id * nElem / num_cpu_threads;
        int threadNum = nElem / num_cpu_threads;
        kernelAddConstant<<<gpu_blocks, gpu_threads>>>(pData, startIdx, threadNum);

        // memory copying
    }
```

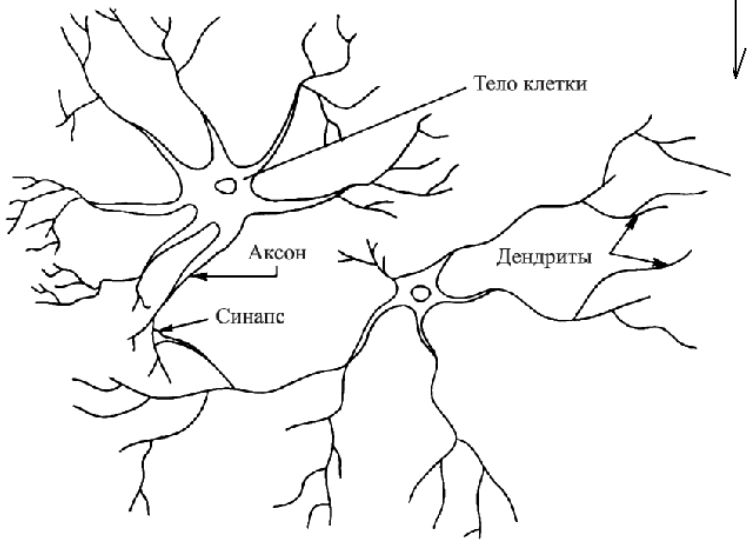
Multi-GPU

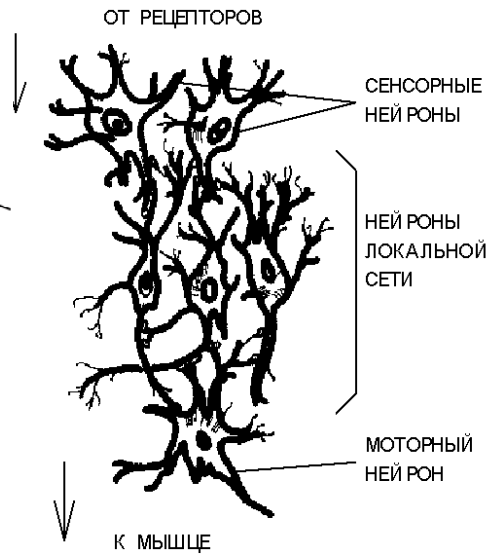
- Разные задачи
- Разделение задачи по данным
  - Обработка изображения
  - Финансовое моделирование
- Разделение по времени
  - Физическое моделирование

Нейронные сети

- Биологическая модель:
  - Мозг состоит и нервных клеток – *нейронов*
  - *Нейрон* выполняет прием, обработку и передачу информации



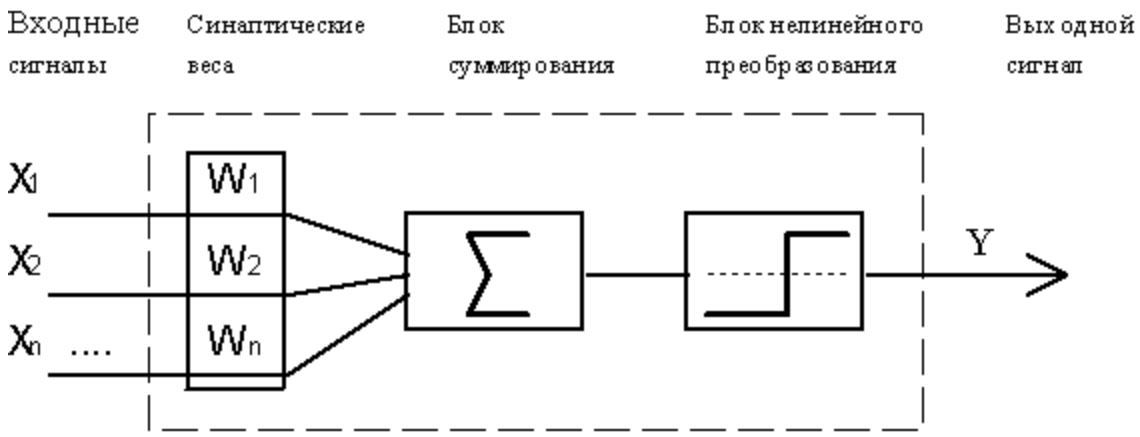




Нейронные сети

- Структура основных нейронных сетей генетически предопределена
- Сеть может меняться
  - Например при отмирании нейронов
  - Нейроны конкурируют за синаптические участки

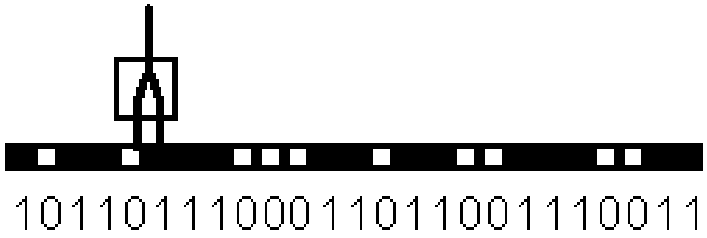
Искуственные Нейронные Сети



- Модель работы нейрона
- Пороговая функция:

$$Y = f(net) = \begin{cases} 1, & net > \Theta \\ 0, & net \le \Theta \end{cases}$$

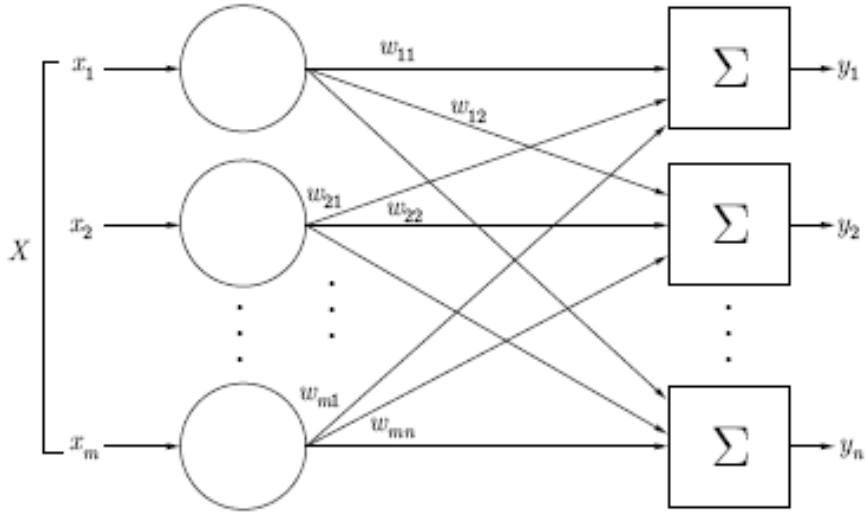
Искуственные Нейронные Сети



- Нейронная сеть определяющая переход от светлого квадрата к темному
  - 2 входа
  - Результат можно задать таблицей:
  - Веса W={1, -1} Θ=0

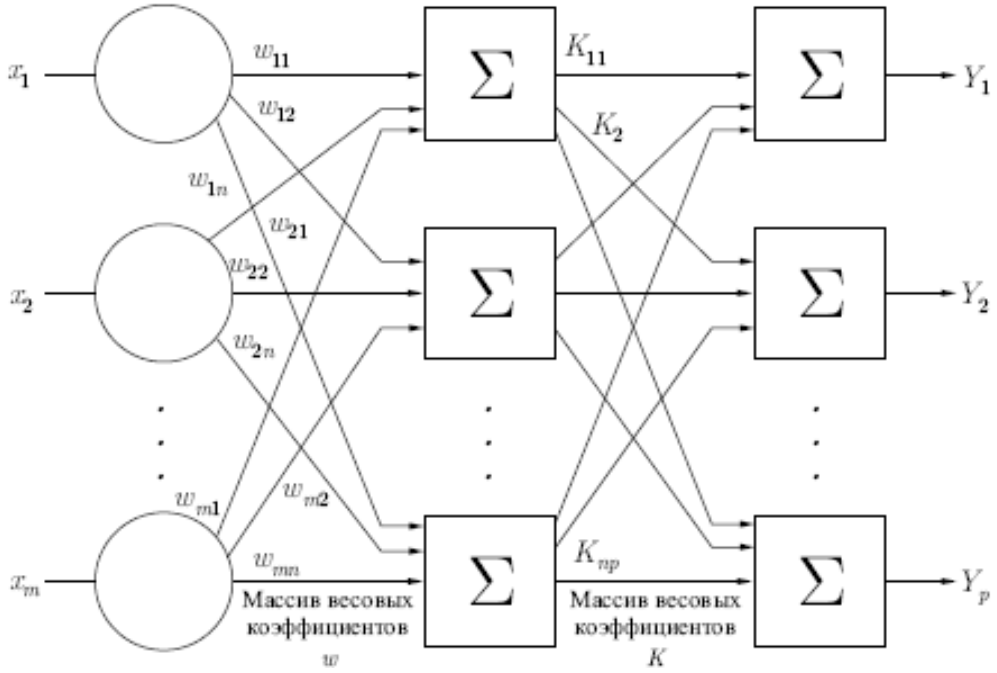
X1	X2	Y
0	0	0
0	1	0
1	0	1
1	1	0

Искуственные Нейронные Сети



- Простейшие нейронные сети – это набор нейронов
- Матрица W={wij} – матрица связей между *i*-м входом и *j*-м нейроном.

Искуственные Нейронные Сети



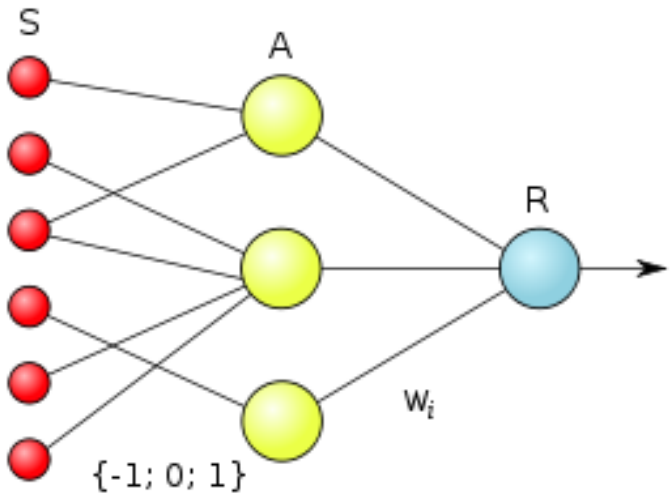
- Можно последовательно соединять, но необходима нелинейная активационная функция

Обучение

- Цель обучения – получить матрицу коэфициентов W
- Обычно рассматриваются пары (X, Y) – входных-выходных (правильных) данных
- При заданных парах можно посчитать функцию ошибки

$$E = E(W) = \sum_{\alpha} \left\| F(X^{\alpha}; W) - Y^{\alpha} \right\| = \sum_{\alpha} \sum_{i} [F_i(X^{\alpha}; W) - Y_i^{\alpha}]^2$$

Перцептрон



- S – вход (сенсор)
- A – ассоциативные элементы
- R – реагирующие элементы

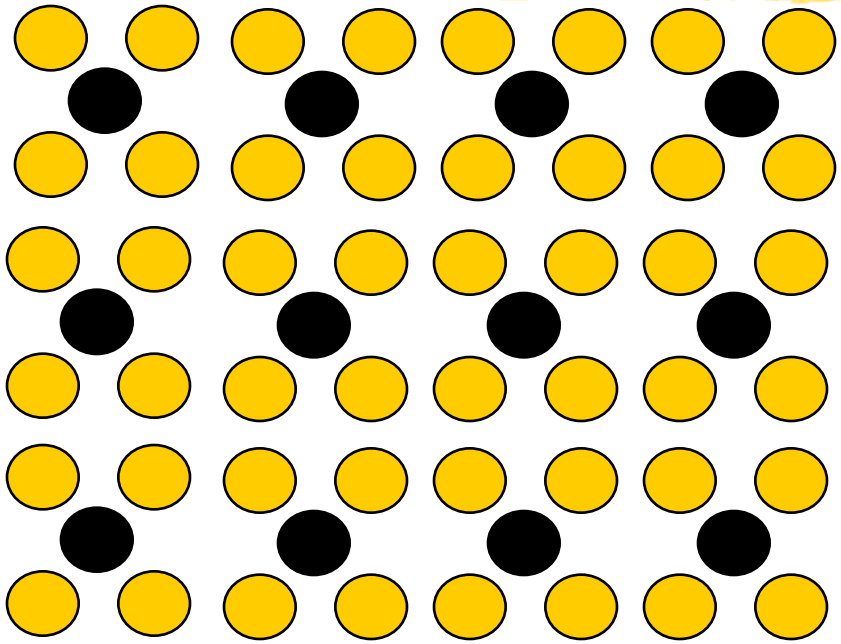
Обучение Перцептрона

- Начальные веса (W(t=0)) считаются случайными
- Сеть принимает на вход пары (x, y) и выдает y’
- Формируется вектор ошибки δ= y-y’
- Вектор весом модифицируется по правилу:$W(t+dt)=W(t)+\eta x(\delta)^T$
- Обучение происходит, пока W не перестанет меняться

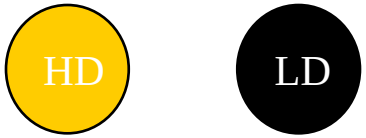
Sony DRC (Digital Reality Creation)

- DRC – запатентованная технология Sony, которая применялась в телевизорах серии WEGA
- Состоит из 2х этапов
  - Нейронная сеть для тренировки фильтра
  - Специальный процессор, который выполняет фильтрацию на лету

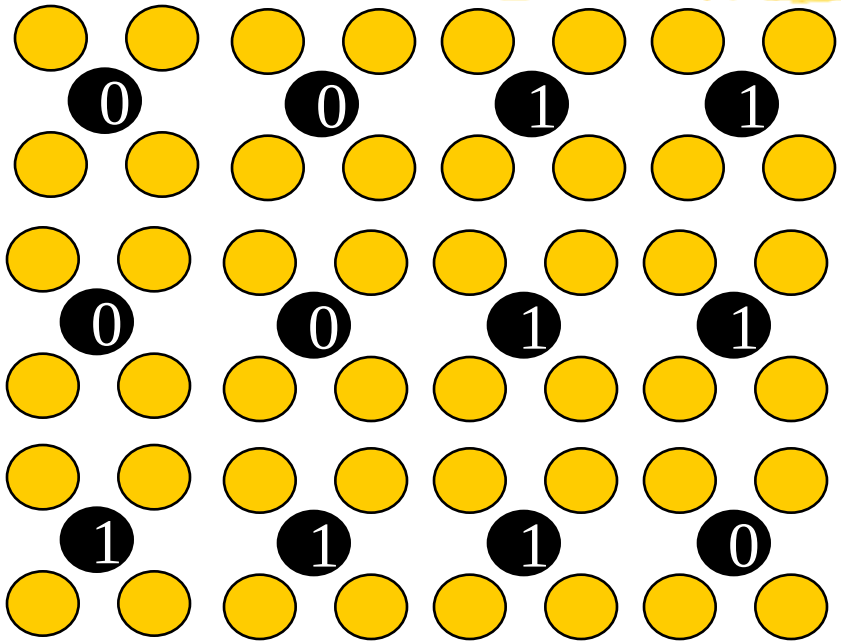
Sony DRC: Тренировка



На вход подается последовательность изображений высокого и низкого разрешениея (HD & LD)



Sony DRC: Тренировка

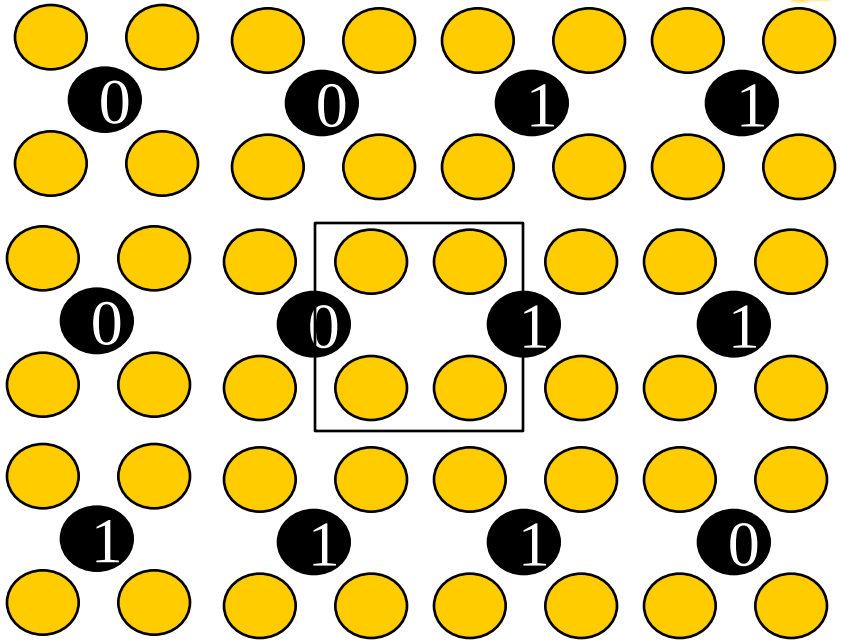


Изображение разбивается на блоки 8x6 (HD) и 4x3 (LD)

LD pixels → Luminance → Threshold: {0, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0}

$$Error(i,j) = HD(i,j) - \sum \alpha(i,j) * LD(i,j)$$

Sony DRC: Тренировка



Расчет идет для 4х центральных пикселей

Метод Наименьших Квадратов

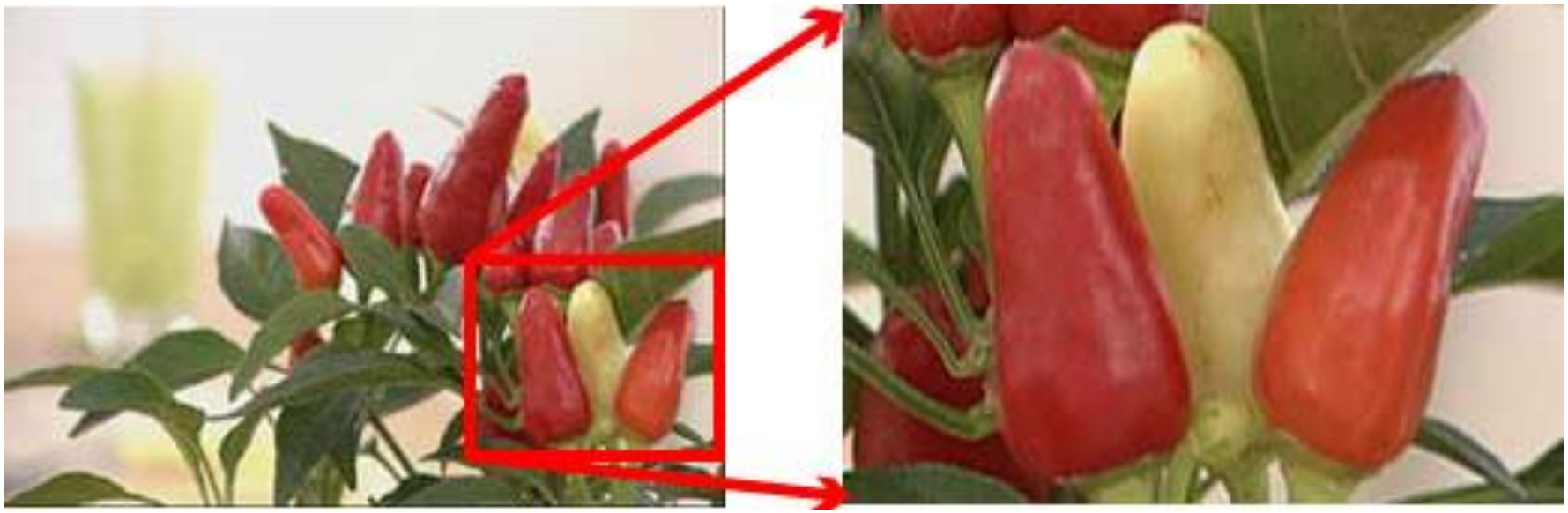
- $HD'_{m,n} = \Sigma a_{ij} * Ld_{ij}$
- $E_{i,j} = HD'_{j,j} - HD_{i,j}$ – Апроксимация Ошибки
- $SSE = \Sigma\Sigma\ sqr(E_{i,j})$; // Summed Square Error
- $a = Arg\ min(SSE)$;
- $Ca = P$
- $\partial(SSE) / \partial a = 0.0$
- $a = (C^T C)^{-1} C^T P$

Sony DRC

- Достоинства:
  - Алгоритм сводится к простой фильтрации на этапе реконструкции изображения
  - 2^{12} разных блоков
    - Для каждого блока свои значения фильтра
- Недостатки:
  - Запатентован Sony

Sony DRC: Тренировка

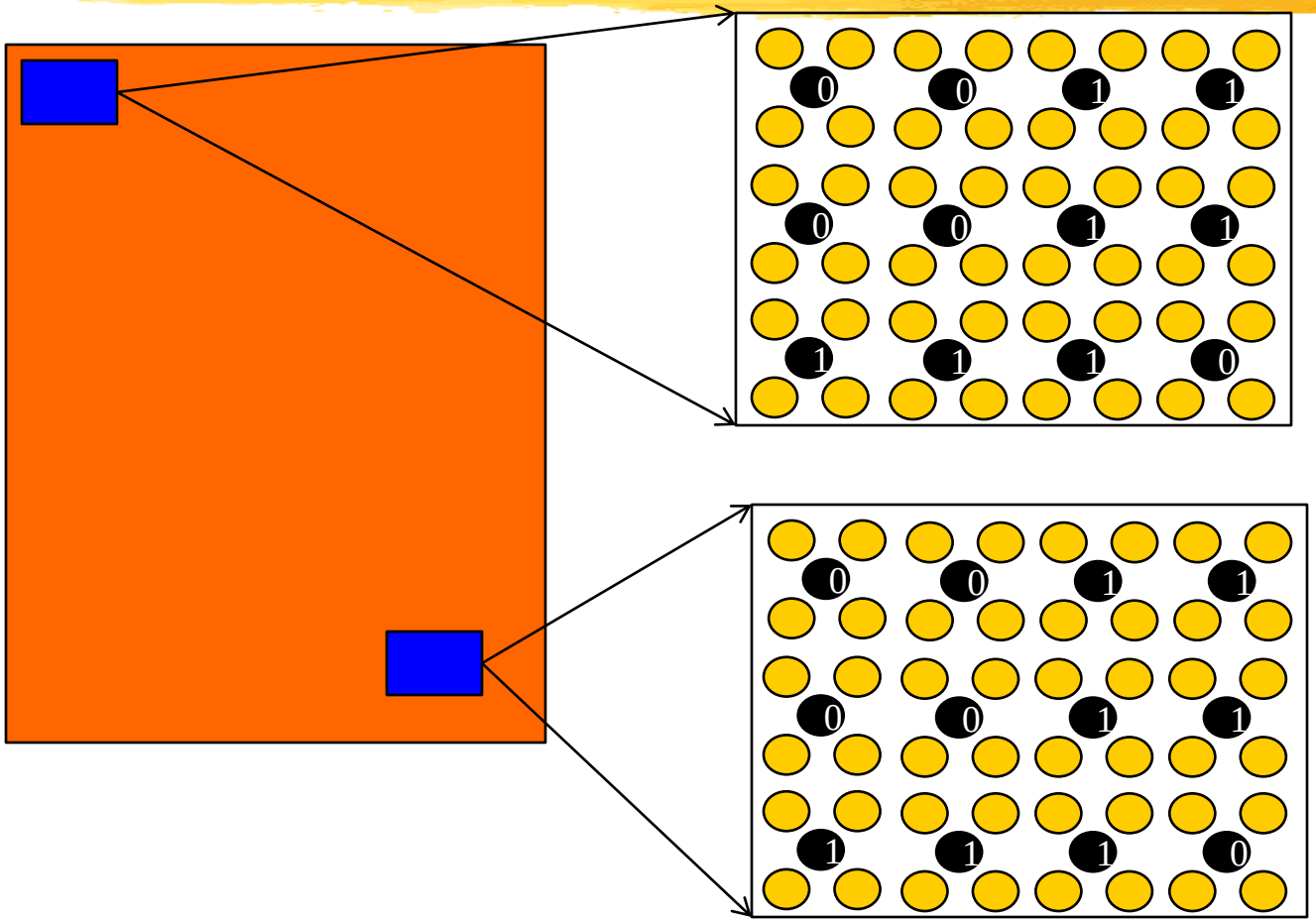
Примеры:



CUDA и DRC

- Как видно из предыдущего примера, есть одна проблема:
  - После классификации блока (получение 12битного слова) необходимо добавить данные блока в соответствующее хранилище данных

CUDA и DRC



CUDA и DRC

- Получаем набор уравнений для одного класса
- Нужно разместить их в памяти
  - Множество потоков работают одновременно на лету
  - Можно использовать *atomic* операции
    - Медленные, т.к. сериализуют доступ в память
  - Либо использовать Scan

Prefix Sum (Scan)

- Определение:

  Префиксная сумма:

  $$[a_0, a_1, \ldots, a_{n-1}]$$

  Возвращает упорядоченный набор

  $$[0, a_0, (a_0 \oplus a_1), \ldots, (a_0 \oplus a_1 \oplus \ldots \oplus a_{n-2})].$$

- Пример:

  [3 1 7 0 4 1 6 3]

  Возвращает набор

  [0 3 4 11 11 15 16 22]

Применение Scan

- Scan – простой и очень полезный примитив
  - Позволяет последовательный код:

```
for(j=1;j<n;j++)
  out[j] = out[j-1] + f(j);
```

  - Сделать параллельным:

```
forall(j) { temp[j] = f(j) };
scan(out, temp);
```

Scan on the CPU

```
void scan( float* scanned, float* input, int length)
{
  scanned[0] = 0;
  for(int i = 1; i < length; ++i)
  {
    scanned[i] = input[i-1] + scanned[i-1];
  }
}
```

- Сложение текущего элемента с предыдущим
- Просто, но последовательно
- Сложность O(n)
- Требует считывать данные на CPU

Вопросы

Организационные вопросы

- 5 мая – занятий нет
- 12 мая
  - Семинар
    - Будут известны результаты всех заданий
    - Можно прийти поспорить
  - Лекция
    - Показ проектов
    - Можно прийти посмотреть