

# **Makalah Komputasi Paralel**

## **“Tugas Proyek Akhir Kelompok”**



Disusun Oleh:

- Fakhri Perdana (1906352060)
- Richardy Lobo' Sapan (1906373954)

Disusun dalam rangka memenuhi tugas

Mata Kuliah Komputasi Paralel

Dosen Pengampu: Alhadi Bustaman, Ph.D

**Fakultas Matematika dan Ilmu Pengetahuan Alam**

**Universitas Indonesia**

**2022**

## **ABSTRAK**

Komputasi Paralel sebagai bentuk komputasi yang lebih efisien memiliki beberapa jenis sistem yang biasa dijalankan untuk komputasi paralel. Pada awalnya, semua pemrosesan dalam komputer hanya diproses oleh CPU. Setelah diciptakannya GPU, graphic rendering akan diproses oleh GPU yang diproses secara parallel. GPU diciptakan pertama kali oleh NVIDIA, yang seiring berjalananya waktu INTEL juga mengeluarkan GPU dengan versinya sendiri. Pada karya ini, akan dibahas mengenai parallel reduction pada OpenMP dan CUDA, serta beberapa contoh implementasi komputasi paralel dalam deep learning yang berhubungan dengan healthcare dan life sciences

Kata Kunci: INTEL, Komputasi Paralel, deep learning

# **DAFTAR ISI**

**HALAMAN JUDUL**

**ABSTRAK**

**DAFTAR ISI**

**BAB I. PENDAHULUAN**

1.1 Latar Belakang

1.2 Rumusan Masalah

1.3 Tujuan Penulisan

1.4 Manfaat Penulisan

**BAB II. PEMBAHASAN**

2.1 Konsep dan proses parallel reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA)

2.2. Contoh makalah tentang teknologi terbaru untuk GPU computing dan Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll)?

2.3 Contoh Implementasi GPU untuk aplikasi Deep Learning?

**BAB III: PENUTUP**

3.1 Kesimpulan

# BAB I

## PENDAHULUAN

### 1. Latar Belakang

Dalam karya ini ini, penulis menjelaskan tentang parallel reduction dan Deep Learning dalam komputasi paralel. OpenMP pada dasarnya memiliki Klausula reduksi OpenMP, di mana Klausula reduksi OpenMP memungkinkan Anda menentukan satu atau lebih variabel thread-private yang tunduk pada operasi reduksi di akhir wilayah paralel. OpenMP menetapkan sebelumnya satu set operator reduksi. Setiap variabel reduksi harus berupa skalar (misalnya, int , long , dan float ).

Pembelajaran mendalam adalah jenis pembelajaran mesin dan kecerdasan buatan (AI) yang meniru cara manusia memperoleh jenis pengetahuan tertentu. Pembelajaran mendalam adalah elemen penting dari ilmu data, yang mencakup statistik dan pemodelan prediktif. Ini sangat bermanfaat bagi ilmuwan data yang bertugas mengumpulkan, menganalisis, dan menafsirkan data dalam jumlah besar; pembelajaran yang mendalam membuat proses ini lebih cepat dan lebih mudah.

Paling sederhana, pembelajaran mendalam dapat dianggap sebagai cara untuk mengotomatiskan analitik prediktif. Sementara algoritme pembelajaran mesin tradisional bersifat linier, algoritme pembelajaran mendalam ditumpuk dalam hierarki yang semakin kompleks dan abstrak.

Untuk memahami pembelajaran yang mendalam, bayangkan seorang balita yang kata pertamanya adalah anjing. Balita belajar apa itu anjing -- dan bukan -- dengan menunjuk ke objek dan mengucapkan kata anjing. Orang tua berkata, "Ya, itu anjing," atau, "Tidak, itu bukan anjing." Saat balita terus menunjuk ke objek, ia menjadi lebih sadar akan fitur yang dimiliki semua anjing. Apa yang dilakukan balita, tanpa menyadarinya, adalah memperjelas abstraksi yang kompleks -- konsep anjing -- dengan membangun hierarki di mana setiap level abstraksi dibuat dengan pengetahuan yang diperoleh dari lapisan hierarki sebelumnya.

Pada makalah ini, penulis akan membahas tentang pengenalan singkat mengenai beberapa hal, diantaranya: (1) Bagaimana konsep dan proses parallel reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA), (2) Bagaimana contoh makalah tentang teknologi terbaru untuk GPU computing dan

Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll)?, dan (3) Bagaimana Contoh Implementasi GPU untuk aplikasi Deep Learning.

## **2. Rumusan Masalah**

Berdasarkan Latar Belakang tersebut, rumusan masalah yang akan digunakan dalam pembentukan makalah ini adalah:

1. Bagaimana konsep dan proses parallel reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA)?
2. Bagaimana contoh makalah tentang teknologi terbaru untuk GPU computing dan Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll)?
3. Bagaimana Contoh Implementasi GPU untuk aplikasi Deep Learning?

## **3. Tujuan Penulisan**

Tujuan yang ingin dicapai dari penulisan makalah ini adalah:

1. Menjelaskan konsep dan proses parallel reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA)
2. Memberikan dan menjelaskan contoh makalah tentang teknologi terbaru untuk GPU computing dan Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll)
3. Memberikan dan menjelaskan Contoh Implementasi GPU untuk aplikasi Deep Learning

## **4. Manfaat Penulisan**

Manfaat yang diharapkan didapat oleh penulisan makalah ini adalah:

1. Dapat memberi ilmu mengenai konsep dan proses parallel reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA)
2. Dapat mengetahui contoh makalah tentang teknologi terbaru untuk GPU computing dan Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll)
3. Dapat mengetahui Contoh Implementasi GPU untuk aplikasi Deep Learning

## **BAB II**

### **PEMBAHASAN**

#### **2.1 Pembahasan konsep dan proses Paralel Reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA).**

##### **Konsep Parallel Reduction**

Reduksi parallel adalah salah satu jenis algoritma PRAM. Reduksi parallel adalah proses memanipulasi data yang disimpan dalam register global. Reduksi paralel mengacu pada algoritma yang menggabungkan array elemen yang menghasilkan nilai tunggal sebagai hasilnya. Masalah yang memenuhi syarat untuk algoritma ini termasuk yang melibatkan operator yang bersifat asosiatif dan komutatif. Salah satu di antaranya termasuk penjumlahan array.

PRAM adalah mesin abstrak memori bersama. Seperti namanya, PRAM dimaksudkan sebagai analogi komputasi paralel dengan mesin akses acak (RAM). Dengan cara yang sama, bahwa RAM digunakan oleh perancang algoritma sekuensial untuk memodelkan kinerja algoritmik (seperti kompleksitas waktu), PRAM digunakan oleh perancang algoritma paralel untuk memodelkan kinerja algoritme paralel (seperti kompleksitas waktu, di mana jumlah prosesor diasumsikan biasanya juga dinyatakan). Mirip dengan cara model RAM mengabaikan masalah praktis, seperti waktu akses ke memori cache versus memori utama, model PRAM mengabaikan masalah seperti sinkronisasi dan komunikasi, tetapi menyediakan sejumlah prosesor (tergantung ukuran masalah).

##### **Proses Parallel Reduction**

###### **Race Condition & Reduction**

Race Condition terjadi ketika beberapa utas membaca dan menulis variabel secara bersamaan, misalnya:

---

```

asum = 0.0d0
 !$omp parallel do shared(x,y,n,asum) private(i)
 do i = 1, n
   asum = asum + x(i)*y(i)
 end do
 !$omp end parallel do

```

---

Hasil acak tergantung pada urutan asum akses utas. penulis membutuhkan beberapa mekanisme untuk mengontrol akses.

Menjumlahkan elemen array adalah contoh operasi reduksi.

$$S = \sum_{j=1}^N A_j = \sum_{j=1}^{\frac{N}{2}} A_j + \sum_{j=\frac{N}{2}+1}^N A_j = B_1 + B_2 = \sum_{j=1}^2 B_j$$

OpenMP menyediakan dukungan untuk reduksi umum dalam region paralel dan loop dengan klausula reduksi sebagai berikut.

### **reduction(operator:list)**

Melakukan pengurangan pada variabel (skalar) dalam daftar

- Variabel pengurangan pribadi dibuat untuk hasil parsial setiap utas
- Variabel reduksi pribadi diinisialisasi ke nilai awal operator
- Setelah wilayah paralel, operasi pengurangan diterapkan ke variabel pribadi dan hasilnya digabungkan ke variabel bersama

Berikut adalah operator reduksi yang ada dalam bahasa C/C++ dan Fortran.

## Reduction operators in C/C++

| Operator | Initial value | Bitwise Operator | Initial value |
|----------|---------------|------------------|---------------|
| +        | 0             | &                | $\sim 0$      |
| -        | 0             |                  | 0             |
| *        | 1             | $\wedge$         | 0             |
| $\&\&$   | 1             |                  |               |
| $\ $     | 0             |                  |               |

## Reduction operators in Fortran

| Operator | Initial value | Bitwise Operator | Initial value |
|----------|---------------|------------------|---------------|
| +        | 0             | .iand.           | all bits on   |
| -        | 0             | .ior.            | 0             |
| *        | 1             | .ieor.           | 0             |
| max      | least         |                  |               |
| min      | largest       |                  |               |
| .and.    | .true.        |                  |               |
| .or.     | .false.       |                  |               |
| .eqv.    | .true.        |                  |               |
| .neqv.   | .false.       |                  |               |

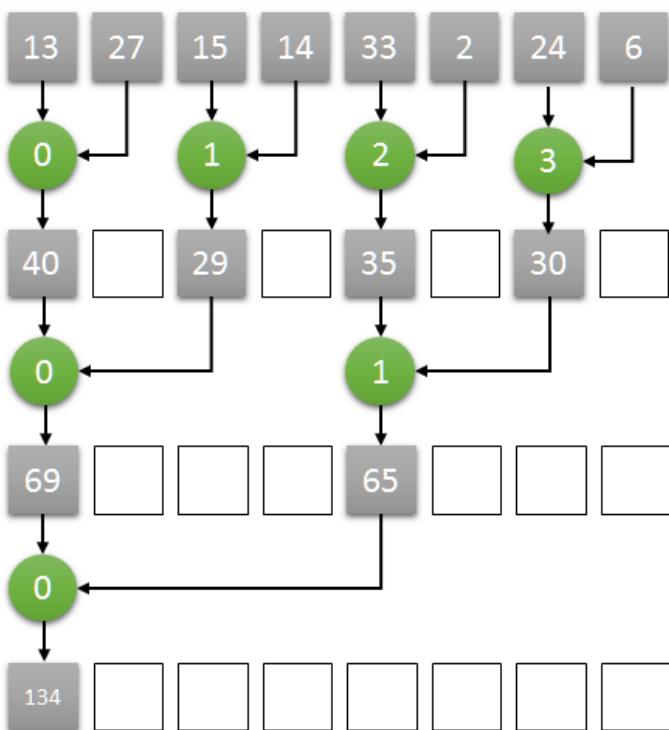
Berikut adalah contoh penggunaan parallel reduction untuk menghindari Race Condition.

## Race condition avoided with reduction clause

```
!$omp parallel do shared(x,y,n) private(i) reduction(+:asum)
do i = 1, n
    asum = asum + x(i)*y(i)
end do
 !$omp end parallel do
```

```
#pragma omp parallel for shared(x,y,n) private(i) reduction(+:asum)
for(i=0; i < n; i++) {
    asum = asum + x[i] * y[i];
}
```

### Proses Parallel Reduction dengan CUDA



Berikut adalah ide utamanya:

- Dengan asumsi N sebagai jumlah elemen dalam array, kita memulai N/2 utas, satu utas untuk setiap dua elemen
- Setiap utas menghitung jumlah dari dua elemen yang sesuai, menyimpan hasilnya pada posisi yang pertama.
- Secara iteratif, setiap langkah:

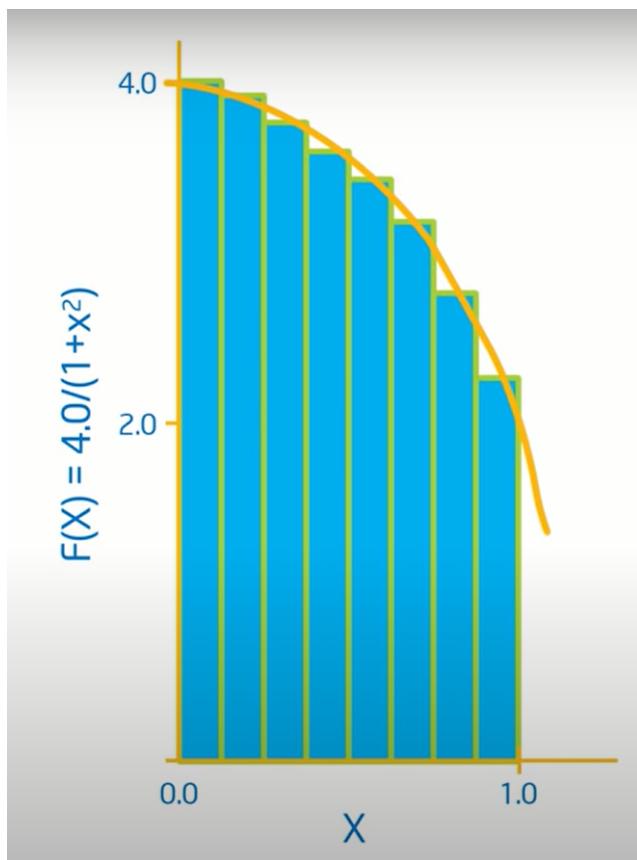
jumlah utas dibelah dua (misalnya, dimulai dengan 4, lalu 2, lalu 1)

menggandakan ukuran langkah antara dua elemen yang sesuai (dimulai dengan 1, lalu 2, lalu 4)

- Setelah beberapa iterasi, hasil akhir dari reduksi akan disimpan di elemen pertama array

## Implementasi dalam OpenMP

### Simulasi OpenMP: Menghitung Pi (integral)



$$\int_0^1 \frac{4.0}{(1+X^2)} dx = \pi$$

Akan dilakukan aproksimasi nilai pi dengan menghitung integral di atas. Berikut adalah percobaan untuk sekuensial dan paralel (reduction) untuk mendekati masalah tersebut.

## 1. Sekuensial

```
10 #include <stdio.h>
11 #include <omp.h>
12 static long num_steps = 100000000;
13 double step;
14 int main ()
15 {
16     int i;
17     double x, pi, sum = 0.0;
18     double start_time, run_time;
19
20     step = 1.0/(double) num_steps;
21
22
23     start_time = omp_get_wtime();
24
25     for (i=1;i<= num_steps; i++) {
26         x = (i-0.5)*step;
27         sum = sum + 4.0/(1.0+x*x);
28     }
29
30     pi = step * sum;
31     run_time = omp_get_wtime() - start_time;
32     printf("\n pi with %ld steps is %lf in %lf seconds\n",num_steps,pi,run_time);
33
34 }
```

```
"C:\Users\A\Desktop\Serial pi calc\bin\Debug\paralel pi calc.exe"
pi with 100000000 steps is 3.141593 in 0.463000 seconds
Process returned 0 (0x0)   execution time : 0.513 s
Press any key to continue.
```

Program ini menghitung integral dari fungsi

$$4/(1+x^2)$$

dari 0 ke 1 secara numerik. Nilai integralnya = pi.

Program ini merupakan versi sekuensial. Program menggunakan timer dari openmp.

## 2. Parallel reduction

```

18 #include <stdio.h>
19 #include <omp.h>
20 static long num_steps = 100000000;
21 double step;
22 int main ()
23 {
24     int i;
25     double x, pi, sum = 0.0;
26     double start_time, run_time;
27
28     step = 1.0/(double) num_steps;
29     for (i=1;i<=4;i++){
30         sum = 0.0;
31         omp_set_num_threads(i);
32         start_time = omp_get_wtime();
33 #pragma omp parallel
34 {
35 #pragma omp single
36     printf(" num_threads = %d",omp_get_num_threads());
37
38 #pragma omp for reduction(+:sum)
39     for (i=1;i<= num_steps; i++){
40         x = (i-0.5)*step;
41         sum = sum + 4.0/(1.0+x*x);
42     }
43 }
44     pi = step * sum;
45     run_time = omp_get_wtime() - start_time;
46     printf("\n pi is %f in %f seconds and %d threads\n",pi,run_time,i);
47 }
48

```

"C:\Users\A\Desktop\serial pi calc\bin\Debug\serial pi calc.exe"

```

num_threads = 1
pi is 3.141593 in 0.415000 seconds and 1 threads
num_threads = 2
pi is 3.153437 in 0.839000 seconds and 2 threads
num_threads = 3
pi is 3.140003 in 0.761000 seconds and 3 threads
num_threads = 4
pi is 3.140808 in 0.671000 seconds and 4 threads

```

```

Process returned 0 (0x0)    execution time : 2.728 s
Press any key to continue.

```

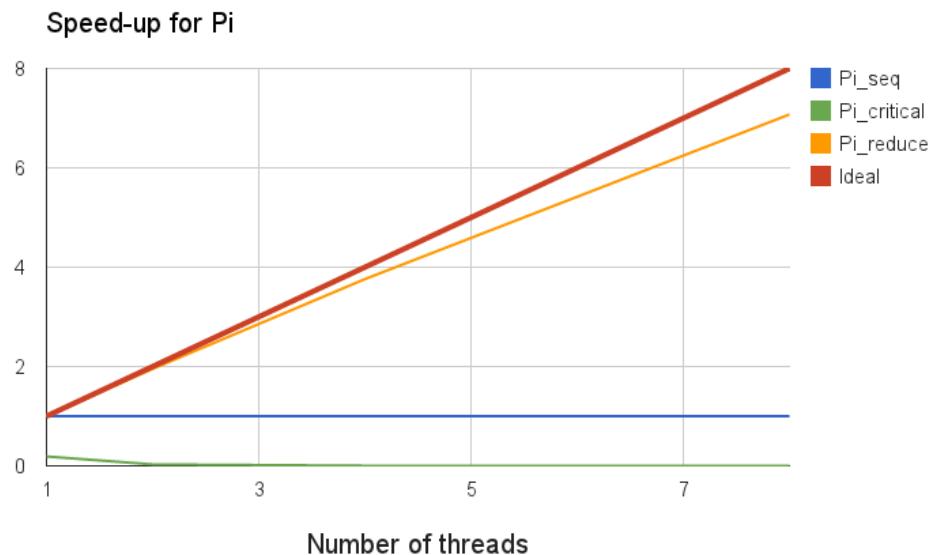
Program ini akan menghitung nilai fungsi  $4/(1+x^2)$  dari 0 ke 1 secara numerik. Nilai integral ini = pi. Program diparalelisisi dengan menambahkan 4 baris,

- (1) Baris yg mengandung omp.h -- file include yg mengandung OpenMP's function prototypes dan constants.
- (2) Pragma utk membuat sekelompok threads
- (3) Pragma yang menyebabkan salah satu thread mencetak jumlah thread yang

digunakan oleh program.

(4) Pragma untuk membagi iterasi loop di antara kelompok threads. Pragma ini mencakup 2 klausa untuk (1) membuat variabel pribadi dan (2) menyebabkan thread menghitung jumlahnya secara lokal dan kemudian menggabungkan jumlah lokalnya menjadi satu nilai global.

## Perbandingan



Speed-up yang ideal adalah jumlah thread, misalnya jika aplikasi bekerja dengan 4 thread, speed-up yang ideal dalam hal ini adalah 4.

Paralelisasi dengan klausa reduksi hampir mencapai speed up ideal, sebaliknya dengan speed up sekuensial lebih buruk. Tampak jelas bahwa dengan paralelisasi, efektivitas dan produktivitas dapat lebih meningkat dibandingkan program sekuensial

Selain cara di atas, parallel reduction pada OpenMP juga dapat dilakukan dengan cara sebagai berikut.

- **Approximation of PI hand-crafting the #pragma omp reduction**

```

h = 1.0 / n;

#pragma omp parallel private(x) shared(n, h)
{
    double thread_area = 0;                                // Private / local variable

    #pragma omp for
    for (i = 1; i <= n; i++)
    {
        x = h * (i - 0.5);
        thread_area += (4.0 / (1.0 + x*x));
    }

    #pragma omp atomic
    area += thread_area;                                // Applies the reduction manually
    // All threads aggregate into area
}

pi = h * area;
return 0;
}

```

Utas muncul dalam paralel #pragma omp. Setiap utas akan memiliki utas\_area independen/pribadi yang menyimpan sebagian tambahannya. Loop berikut didistribusikan di antara utas menggunakan #pragma omp for. Dalam loop ini, setiap thread menghitung thread\_areanya sendiri dan setelah loop ini, kode secara berurutan menggabungkan area secara atom melalui #pragma omp atomic.

- **Approximation of PI using #pragma omp reduction clause**

```

h = 1.0 / n;
#pragma omp parallel for private(x) shared(n, h) reduction(+:area)
for (i = 1; i <= n; i++)
{
    x = h * (i - 0.5);
    area += (4.0 / (1.0 + x*x));
}
pi = h * area;
.
.
.

```

Dalam contoh ini, setiap utas mengeksekusi subset dari jumlah iterasi. Setiap utas memiliki salinan area pribadi lokalnya dan pada akhir wilayah paralel mereka semua menerapkan operasi penambahan (+) sehingga menghasilkan nilai akhir untuk area.

- **Approximation of PI using reductions based on #pragma atomic**

```

int main()
{
    h = 1.0 / n;
    #pragma omp parallel for private(x) shared(n, h, area)
    for (i = 1; i <= n; i++)
    {
        x = h * (i - 0.5);
        #pragma atomic
        area += (4.0 / (1.0 + x*x));
    }
    pi = h * area;

    return 0;
}

```

Dalam contoh ini, setiap utas mengeksekusi subset dari jumlah iterasi dan mereka terakumulasi secara atom ke dalam area variabel bersama, yang memastikan bahwa tidak ada pembaruan yang hilang. Kita dapat menggunakan #pragma atomik di sini karena operasi yang diberikan (+=) dapat dilakukan secara atom, yang menyederhanakan keterbacaan dibandingkan dengan penggunaan #pragma omp kritis.

- Approximation of PI using reductions based on #pragma omp critical

```

h = 1.0 / n;
#pragma omp parallel for private(x) shared(n, h, area)
for (i = 1; i <= n; i++)
{
    x = h * (i - 0.5);
    #pragma omp critical
    {
        area += (4.0 / (1.0 + x*x));
    }
    pi = h * area;
}

```

Dalam contoh ini, setiap utas mengeksekusi subset dari jumlah iterasi dan mereka terakumulasi secara atom ke dalam area variabel bersama, yang memastikan bahwa tidak ada pembaruan yang hilang.

## Implementasi Menggunakan CUDA

Disini kita menggunakan CUDA dengan bantuan Google COLAB, dengan menambahkan fungsi “%%cu”, unutk menjalankan CUDA nya.

```
%%cu
#include "cuda_runtime.h"
#include "device_launch_parameters.h"

#include <iostream>
#include <numeric>
using namespace std;

__global__ void sum(int* input)
{
    const int tid = threadIdx.x;

    auto step_size = 1;
    int number_of_threads = blockDim.x;

    while (number_of_threads > 0)
    {
        if (tid < number_of_threads) // still alive?
        {
            const auto fst = tid * step_size * 2;
            const auto snd = fst + step_size;
            input[fst] += input[snd];
        }
    }
}
```

```
step_size <= 1;
number_of_threads >= 1;
}

int main()
{
    const auto count = 8;
    const int size = count * sizeof(int);
    int h[] = {13, 27, 15, 14, 33, 2, 24, 6};

    int* d;

    cudaMalloc(&d, size);
    cudaMemcpy(d, h, size, cudaMemcpyHostToDevice);

    sum <<<1, count / 2 >>>(d);

    int result;
    cudaMemcpy(&result, d, sizeof(int), cudaMemcpyDeviceToHost);

    cout << "Sum is " << result << endl;

    getchar();
}
```

```
cudaFree(d);
delete[] h;

return 0;
```

```
Sum is 134
```

**2.2 Makalah tentang teknologi terbaru untuk GPU computing dan Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll).**

**Sumber Makalah GPU Computing :**

AvadheshPratapSingha, M. Tech Scholar, MANIT Bhoapl, MP, DhirendraPratapSingh India.Assistant Professor, MANIT Bhoapl, MP, India

Implementation of K-shortest path algorithm in GPU using CUDA

© 2014 © 2015 The Authors. Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license The Authors. Published by Elsevier B.V. (<http://creativecommons.org/licenses/by-nc-nd/4.0/>). Peer-review under responsibility of scientific committee of International Conference on Computer, Communication and Convergence (ICCC 2015)

<https://www.sciencedirect.com/science/article/pii/S1877050915006122>

Ringkasan :

**Abstrak**

Algoritma K-shortest path merupakan generalisasi dari algoritma shortest path. Jalur terpendek K digunakan di berbagai bidang seperti masalah penyelarasan urutan dalam bioinformatika molekuler, perencanaan gerak robot, pencarian jalur dalam jaringan gen di mana kecepatan untuk menghitung jalur memainkan peran penting. Implementasi paralel adalah salah satu cara terbaik untuk memenuhi kebutuhan aplikasi ini. Algoritma paralel berbasis GPU dikembangkan untuk menemukan k jumlah jalur terpendek dalam graf besar berarah berbobot positif. Dalam penghitungan jalur terpendek, pengulangan simpul tidak diperbolehkan. Algoritma yang diimplementasikan menghitung k-shortest path antara dua pasang simpul dari suatu graf dengan n simpul dan m simpul. Pendekatan ini didasarkan pada algoritma Yen untuk menemukan k-shortest loopless path. Kami menerapkan algoritme kami di GPU Nvidia menggunakan Compute Unified Device Architecture (CUDA). Makalah ini menyajikan analisis komparatif antara implementasi Algoritma Yen berbasis CPU dan GPU. Pendekatan kami mencapai kecepatan 6 kali dibandingkan dengan algoritma serial.

## Representasi GPU

Representasi graf untuk mengakses graf, berperan penting dalam menjalankan waktu algoritma. Secara konvensional, ada dua cara untuk merepresentasikan grafik, matriks ketetanggaan dan daftar ketetanggaan. Matriks ketetanggaan menghabiskan banyak memori dalam kasus grafik yang jarang. Representasi daftar ketetanggaan adalah cara terbaik untuk merepresentasikan graf sparse. Pada perangkat GPU, CUDA mengakses memori secara array sehingga karena ukuran edge list yang berbeda sulit untuk menggunakan Adjacency list. Haris et. al.<sup>5</sup> dan Dhirendraet. al.<sup>9</sup> memberikan representasi daftar kedekatan yang dimodifikasi. Menurut Dhirendraet. al.<sup>9</sup> menggunakan tiga array  $V_a$ ,  $E_a$  dan  $E_w$ . Titik-titik dari graf  $G(V,E,W)$  direpresentasikan sebagai larik  $V_a$ .  $V_a$  digunakan untuk menyimpan indeks awal dari daftar kedekatan sendiri di  $E_a$ .  $E_a$  array berukuran sama dengan jumlah edge, digunakan untuk menyimpan nomor vertex yang terhubung ke vertex ke- $i$  dari  $V_a$ . Dalam array  $E_a$  setiap entri dalam range nilai ke- $i$  sampai nilai  $(i+1)$  dari  $V_a$ , terhubung ke- $i$  simpul  $V_a$  untuk semua  $i$  di  $V_a$ .  $E_w$  digunakan untuk menyimpan bobot tepi yang sesuai dengan  $E_a$ .

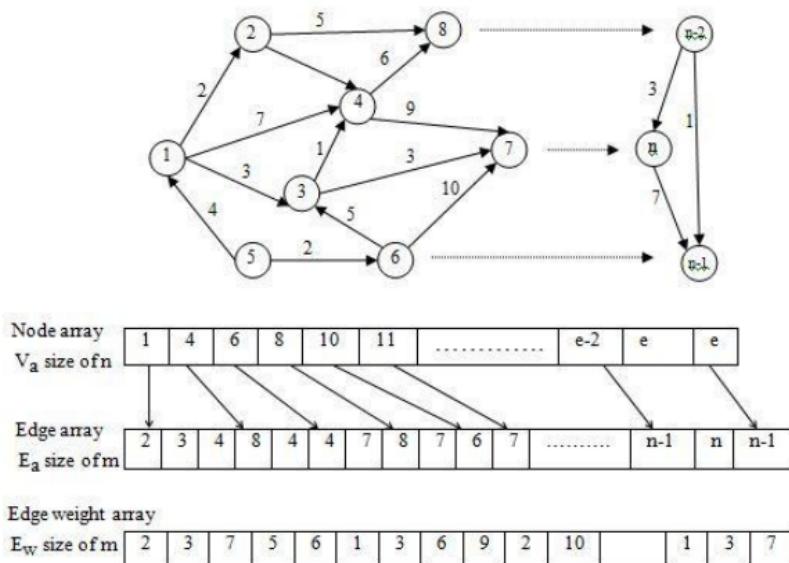


Fig. 3. Graph Representation

Menurut Dhirendraet. al.<sup>9</sup> dalam waktu praproses  $E_w$  menyimpan bobot dalam urutan terpendek dalam kisaran nilai ke- $i$  hingga  $(i+1)$  nilai  $V_a$ , terhubung ke simpul ke- $i$   $V_a$  untuk

semua i di Va. Pemrosesan terurut ini berguna dalam perhitungan jalur terpendek. Gambar 3 menunjukkan representasi grafik dalam CUDA. Dengan cara ini

representasi kita dapat dengan mudah menghitung derajat keluar dari setiap simpul. Derajat keluar dari simpul i sama dengan selisih indeks ke-i dari Va dan indeks (i+1) dari Va. Kita telah menambahkan satu lagi array Es dengan ukuran |E| untuk menyimpan simpul awal dari setiap tepi. Representasi grafik ini dilakukan sedemikian rupa sehingga akan mudah diakses di GPU dan membantu mengurangi waktu akses algoritma yang sedang berjalan. Semua array yang ditentukan ini diinisialisasi dalam pra-pemrosesan grafik.

### **Struktur data untuk mewakili Path**

Biarkan Z menjadi array yang menyimpan informasi yang terkait dengan setiap jalur terpendek yang dihitung. Dalam makalah ini kami menggunakan dua larik, satu untuk kumpulan jalur hasil (Zresult) dan satu lagi untuk kumpulan jalur lainnya (Zrest). Ukuran Zresult sama dengan K (jumlah jalur terpendek). Jika jalur yang dihitung kurang dari K maka beberapa bidang array adalah nihil. Setiap elemen array menyimpan informasi berikut tentang jalur P yang diwakilinya. Pointer ke array Path yang menyimpan jalur terpendek antara dua node. Setiap elemen array Path menyimpan dua informasi yaitu node dan bobot dari sumbernya. Nilai indeks jalur induk dari mana jalur tersebut dihasilkan (jika jalur pertama atau tidak bergantung pada jalur yang dihitung sebelumnya, maka nilai bidang ini negatif). Indeks simpul pengalihan dari jalur induk dari mana jalur baru dihitung (jika simpul sumber adalah simpul pengalih maka nilai bidang ini negatif). Panjang jalur yang sesuai. Struktur data ini digunakan dalam meminimalkan panggilan fungsi dari algoritma jalur terpendek dibandingkan dengan algoritma yen. Node pengalihan memastikan bahwa tidak ada jalur terpendek duplikat di Zrest karena kami mulai menghitung jalur baru setelah menghapus tepi dalam grafik setelah indeks simpul pengalihan dan seterusnya sehingga tidak ada jalur duplikat. Struktur data ini membantu melacak jalur induk dimana jalur baru dihasilkan sehingga tidak perlu membandingkan setiap jalur untuk tepi umum di jalur.

### **Algoritma dan implementasi paralelnya**

Modifikasi algoritma Yen menggunakan algoritma Dijkstra Algoritma Dijkstra pada dasarnya digunakan untuk perhitungan single source shortest path1. Terdapat berbagai implementasi paralel 2,4,5,6,7,8,9 dari algoritma Dijkstra di GPU menggunakan CUDA. Dengan beberapa modifikasi, algoritma Dijkstra dapat digunakan untuk menghitung jalur

terpendek antara dua simpul. Algoritma Yen secara internal menggunakan perhitungan jalur terpendek dimana kita akan menggunakan algoritma paralel Dijkstra<sup>9</sup>. Algoritma Yen pada dasarnya digunakan untuk menghitung k-shortest simple path pada graf berbobot berarah. Algoritma Yen juga dapat digunakan untuk menghitung jalur terpendek pada graf tak berarah tetapi tidak perlu menghitung banyak jalur terpendek yang mengarah ke sana dalam waktu yang lebih lama. Katoahet al. 22 memberikan algoritma yang efisien untuk graf tak berarah.

---

**Algorithm 1: Modified Yen's algorithm (Graph G (V, E, W), Source node, Destination node , K\_no)**


---

Create two list for the Result path set RESULT and rest path set REST

**Begin**

- [1] Calculate first shortest path using DIJKASTRA algorithm.
- [2] Add the first shortest path in the  $Z_{result}$ .
- [3] **for** k=2 to K\_no
- [4] **for** each edge of the (k-1)th path( $s, v_1, v_2, v_3, \dots, v_n$ ) of  $Z_{result}$
- [5] Remove each edge ( $v_i, v_{i+1}$ ) from graph corresponding  $Z_{result}$  to where same sub path from s to  $v_i$
- [6] Apply shortest path algorithm for  $v_i$  to t
- [7] Combine path s to  $v_i$  and new path from  $v_i$  to t
- [8] Store the calculated path in  $Z_{rest}$
- [9] **End of for**
- [10] Take minimum path from the REST path list and store in  $Z_{result}$ .
- [11] **End of for**

**End**

---



---

**Algorithm 2: Dijkastra\_SSSP (Node\_weight, Mask, res\_node, Edge\_start, Edge, Weight, S\_node, D\_node)**


---

**Begin**

- [1] **while**( $thre < infinite$  and  $Mask[D\_node] \neq 1$ ) do
- [2]  $thre = infinite$
- [3] CAL\_THRES(Node, Node\_weight, Edge, Weight, Mask,  $thre, infinite$ ) for all nodes of the graph in parallel
- [4] REL\_NODE(Node, Node\_weight, Edge, Weight, Mask,  $thre$ ) for all nodes of the graph in parallel
- [5] **Endwhile**
- [6] FIND\_RESPONSIBLE(Edge\_start, res\_node, edge, edge weight , Node\_weight ) for all edge of the graph in parallel

**End**

---



---

**Algorithm 3: FIND\_RESPONSIBLE (Edge\_start, mask, edge, weight, Node\_weight )**


---

**Begin**

- [1] id = getThreadID
- [2] **if**( $Node\_weight[Edge\_start[id]] + weight[id] == Node\_weight[edge[id]]$ )
- [3] then  $res\_node[edge[id]] = Edge\_start[id]$
- [4] **End if**

**End**

---



---

**Algorithm 4: Yen\_parallel(Graph G (V, E, W), S\_node, D\_node, K)**


---

Create an array Node\_weight of size  $|V|$ , a Boolean array Mask of size  $|V|$ , a array res\_node of size  $|V|$ , a variable  $infinite$  with a very large number assigned to it and a variable  $thre$  to store the threshold value. Create two array  $Z_{result}$  and  $Z_{rest}$ . Size of  $Z_{result}$  is K

---

```

Begin
[1] INITIATE(Node_weight, Mask, S_node) for all nodes of the graph in parallel
[2] thre= 0
[3] Dijkstra_SSSP (Node_weight, Mask, res_node, Edge_start, Edge, Weight, S_node, D_node)
[4] Add the first shortest path in the Zresult in first index and initiate diversion node and parent node value to 0 and -1.
[5] for k=2 to K
[6] foreach edge fromm=diversion node totod_node of (k-1)th path of Zresult.
[7] n= n = Zresult[k-1].
[8] while n = parent path is non-negative and m == Zresult[diversion node].
[9] remove edge (vm, vm+1) from graph.
[10] n = Zresult[n].
[11] End of while
[12] Dijkstra_SSSP (Node_weight, Mask, res_node, Edge_start, Edge, Weight, m, D_node)
[13] Store the calculated path in Zrest.
[14] End of for
[15] find minimum path weight value from Zrest and store in path_w and store index in min.
[16] if(path_w< infinite)
[17] Zresult [k]=Zrest[min] and set Zrest to NULL.
[18] else
[19] exit (there is no more path)
[20] End of for
End

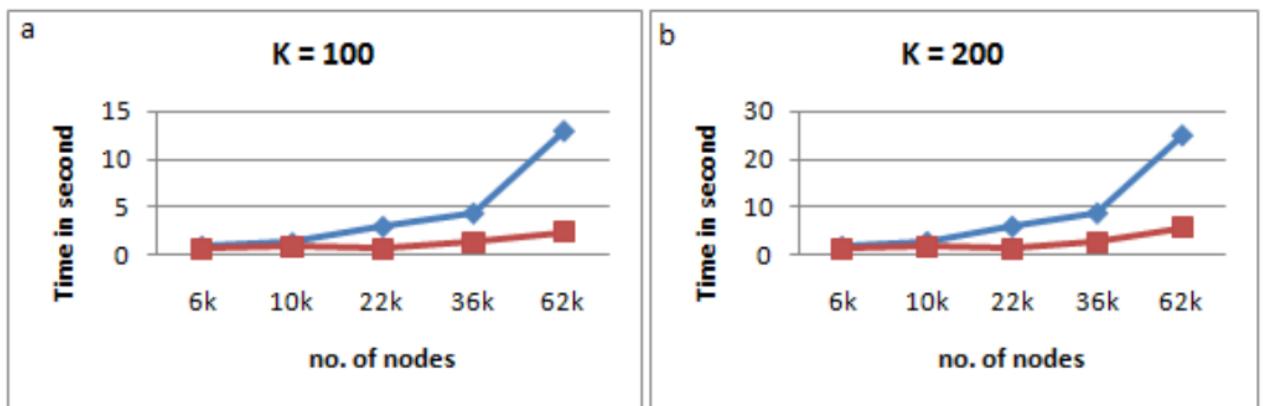
```

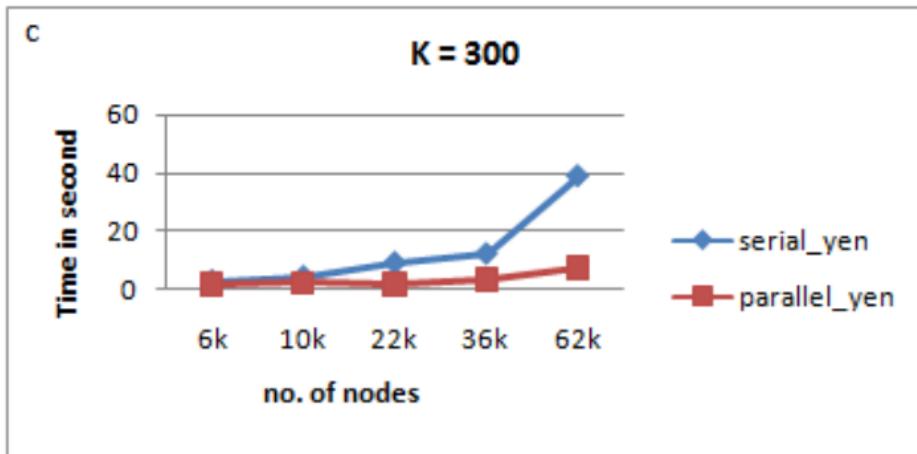
---

## Analisis Performanya

Table 1. Experimental Setup

| Specification      | Version / Detail            |
|--------------------|-----------------------------|
| CUDA Version       | 5.0                         |
| Nvidia GPU         | Tesla C2075                 |
| Compute Capability | 2.1                         |
| Cores              | 448                         |
| Multiprocessor     | 14                          |
| CPU Processor      | 2 x CPU Intel HEX(6), 2.8Hz |
| RAM                | 24GB                        |
| GPU Memory         | 4GB                         |
| OS                 | Windows 7                   |
| Visual Studio      | Professional 2010           |





Gambar (a), gambar (b) dan gambar (5) menunjukkan hasil dari algoritma serial yen (serial\_yen) dan implementasi paralel dari algoritma yen (parallel\_yen) pada setup yang ditentukan. Hasilnya menunjukkan analisis komparatif dalam grafik dengan 62k dan 1,4 juta tepi. Untuk  $k=100$  parallel\_yen menunjukkan percepatan 6,5 waktu dalam grafik. Derajat rata-rata dari graf berarah yang digunakan adalah 3 sampai 5. Ketika kita menaikkan nilai  $K$ , waktunya hampir sama untuk semua nilai  $K$ . Sebuah graf dengan simpul 22k menunjukkan waktu yang lebih sedikit karena derajat rata-ratanya adalah 6 sampai 7. Jadi kita dapat mengatakan bahwa kepadatan grafik meningkat maka waktu untuk menghitung jumlah  $K$  grafik berkurang.

## Kesimpulan

Dalam makalah ini kami telah merancang dan mengimplementasikan algoritma yen secara efisien menggunakan algoritma paralel Dijkstra. Algoritma jalur terpendek  $K$  diimplementasikan di GPU pertama kali. Kami telah menggunakan struktur data baru yang sangat cocok untuk GPU dan juga mengurangi waktu berjalan algoritme. Dengan menggunakan struktur data baru panggilan internal jalur terpendek berkurang dan dengan mudah kita dapat mengidentifikasi bagaimana grafik dimodifikasi sementara untuk mendapatkan jalur terpendek baru

dari yang sebelumnya. Akhirnya kami mendapatkan pohon jalur terpendek dengan bobot jalur yang sesuai. Kami telah menguji algoritme kami untuk grafik yang berbeda dengan berbagai nilai  $K$ . Kami memiliki kecepatan 6x lebih cepat dibandingkan dengan implementasi

serial yen. Untuk mendapatkan kinerja yang lebih, kami dapat menerapkan algoritme ini beberapa GPU sekaligus sehingga kami dapat menetapkan beberapa grafik yang dimodifikasi di GPU yang berbeda sehingga kami dapat menemukan beberapa jalur terpendek. Menggunakan Beberapa GPU, lebih banyak keuntungan cabbe paralelisasi. Kami juga dapat menggunakan hierarki memori GPU untuk mengurangi waktu akses memori.

### **Sumber Makalah Deep Learning:**

Sendak M, Ratliff W, Sarro D, Alderton E, Futoma J, Gao M, Nichols M, Revoir M, Yashar F, Miller C, Kester K, Sandhu S, Corey K, Brajer N, Tan C, Lin A, Brown T, Engelbosch S, Anstrom K, Elish M, Heller K, Donohoe R, Theiling J, Poon E, Balu S, Bedoya A, O'Brien C

Real-World Integration of a Sepsis Deep Learning Technology Into Routine Clinical Care: Implementation Study

JMIR Med Inform 2020;8(7):e15182

URL: <https://medinform.jmir.org/2020/7/e15182>

DOI: 10.2196/15182

### **Ringkasan:**

### **Latar belakang**

Teknologi yang mendigitalkan dan memanfaatkan sejumlah besar data yang dipasangkan dengan penyelarasan penelitian dan perawatan klinis mengubah perawatan kesehatan. Pembelajaran mesin, seperangkat metode statistik yang dioptimalkan untuk prediksi pada pengamatan baru, merupakan inti dari transformasi ini. Meskipun jalur translasi untuk model prognostik dicirikan dengan baik, beberapa model pembelajaran mesin divalidasi atau dievaluasi secara eksternal dalam praktik klinis. Upaya terisolasi menunjukkan dampak klinis dari teknologi yang sebelumnya divalidasi menunjukkan potensi pembelajaran mesin dalam perawatan kesehatan. Namun, tantangan signifikan tetap ada untuk teknologi pembelajaran mesin agar sepenuhnya tertanam dalam operasi standar sistem pemberian perawatan kesehatan.

Pembelajaran mesin telah dengan cepat diadopsi dalam ilmu biomedis untuk meningkatkan metode prediktif, prognostik, dan diagnostik. Namun, banyak hambatan teknis dan klinis untuk adopsi tetap ada. Pertama, catatan kesehatan elektronik (EHR) seringkali tidak memiliki fungsi asli untuk mengintegrasikan model pembelajaran mesin yang kompleks. Diperlukan investasi yang signifikan dalam infrastruktur. Kedua, bahkan setelah model awalnya diimplementasikan, model pembelajaran mesin dapat menimbulkan biaya pemeliharaan berkelanjutan yang substansial. Ketiga, meskipun beberapa sistem kesehatan membangun dan mengintegrasikan solusi pembelajaran mesin yang dikembangkan sendiri, upaya itu sering dialihdayakan ke tim peneliti atau vendor teknologi. Pemisahan antara operasi dan implementasi serta pemeliharaan model ini menghadirkan tantangan tambahan, karena "kepemilikan rekayasa dari sinyal input terpisah dari kepemilikan rekayasa model yang mengkonsumsinya." Akhirnya, banyak model tidak terintegrasi secara efektif ke dalam alur kerja klinis dengan cara yang meningkatkan perawatan klinis atau hasil.

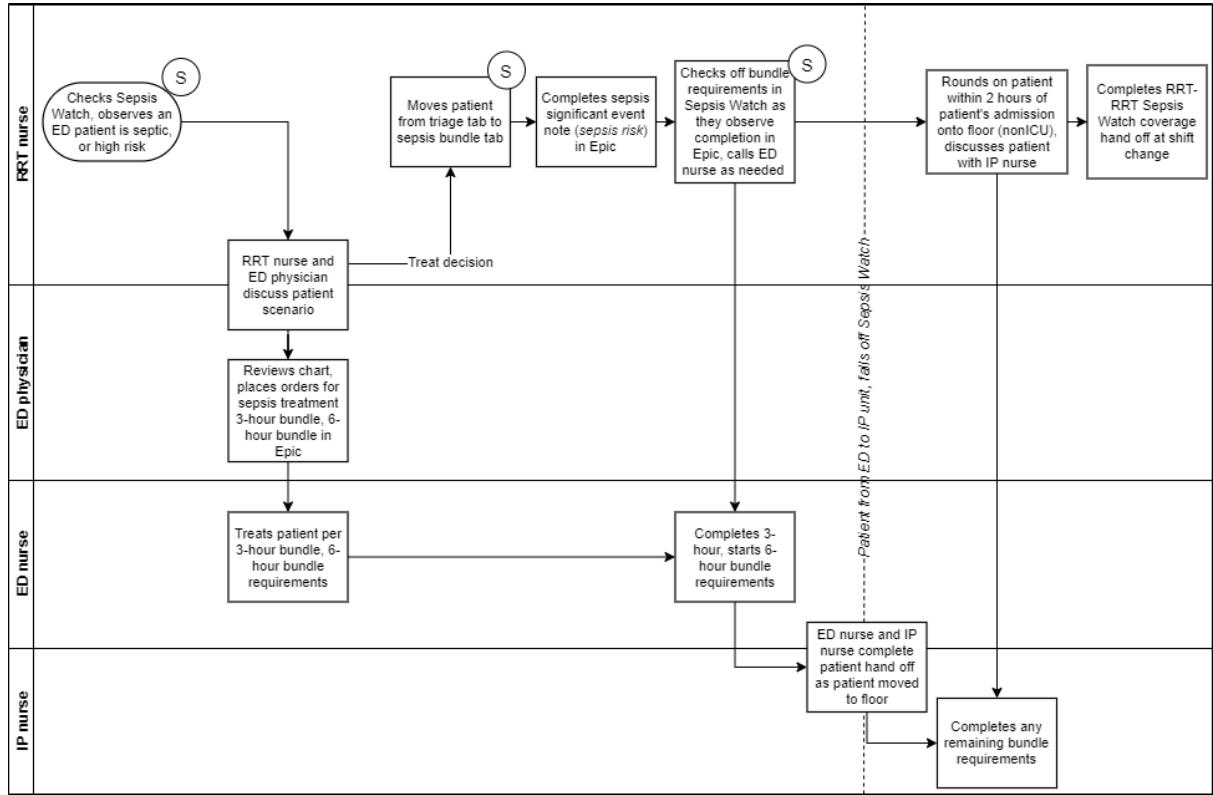
## Sepsis Watch

Di sini, penulis mempelajari detail tentang bagaimana sistem kesehatan mengintegrasikan teknologi pembelajaran mendalam skala penuh pertama ke dalam perawatan klinis rutin. Sebuah kelompok inovasi menghabiskan lebih dari dua tahun dengan mitra di seluruh organisasi untuk meluncurkan solusi pembelajaran mendalam, Sepsis Watch, pada 5 November 2018. Sepsis Watch adalah platform deteksi dan manajemen sepsis yang digunakan oleh dokter untuk meningkatkan kepatuhan terhadap pedoman pengobatan yang direkomendasikan untuk sepsis dan sehingga meningkatkan hasil pasien. Meskipun Sepsis Watch adalah contoh dukungan keputusan klinis pembelajaran mesin (CDS), sistem pembelajaran mendalam memang menimbulkan tantangan implementasi di luar CDS tradisional, seperti yang dirinci di tempat lain. Secara khusus, mekanisme kepercayaan dan akuntabilitas baru harus dikembangkan untuk memastikan bahwa sistem tersebut aman dan dapat diandalka. Pada Tabel berikut, penulis menyajikan 8 langkah yang diperlukan untuk mengintegrasikan Sepsis Watch ke dalam pemberian perawatan rutin dengan sukses. Penulis mengambil pelajaran dari kerangka kerja sistem kesehatan pembelajaran dan praktik terbaik yang dijelaskan sebelumnya untuk pembelajaran mesin yang bertanggung jawab dalam perawatan kesehatan. Tujuan dari naskah ini adalah untuk menggambarkan setiap langkah secara rinci dan menyoroti pembelajaran yang dapat menginformasikan upaya terkait di organisasi lain.

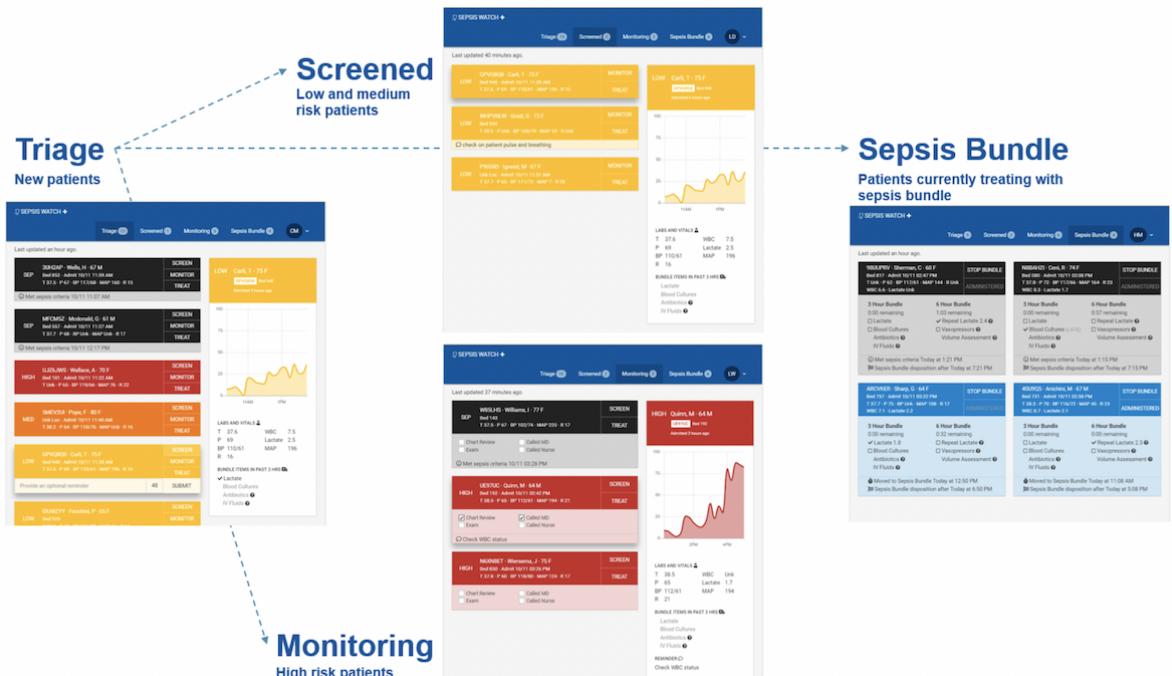
**Table 1.** Steps for integrating machine learning into clinical care. The table includes definitions for the various steps and example tasks and deliverables during the step.

| Step in the process                                      | Definition  | Example tasks and milestones  |
|--|---|---|
| • Problem assessment                                     | <ul style="list-style-type: none"> <li>Understand the root cause of the problem, the magnitude of the problem, where the problem is felt most acutely, who is best positioned to address the problem, and what changes need to occur to empower someone to address the problem</li> </ul> | <ul style="list-style-type: none"> <li>Data analysis to understand the magnitude, setting, and timing of the problem</li> <li>Observe frontline staff in clinical settings where the problem occurs</li> <li>Interview a broad group of stakeholders to understand complexities in addressing the problem</li> </ul>  |
| • Internal and external scans of solutions and workflows | <ul style="list-style-type: none"> <li>Perform due diligence on internal and external tools that attempt to address the problem</li> </ul>  | <ul style="list-style-type: none"> <li>Evaluate technologies and workflows available through current information technology supplier relationships</li> <li>Evaluate technologies on the market sold by external vendors</li> <li>Interview internal stakeholders who have previously attempted to solve the problem</li> </ul>   |
| • Clinical workflow design                               | <ul style="list-style-type: none"> <li>Design clinical workflow that integrates new technology to address the problem</li> </ul>  | <ul style="list-style-type: none"> <li>Gather requirements from frontline staff and leadership</li> <li>Iterate on workflow designs with frontline staff</li> <li>Identify constraints (eg, time and effort) to ensure that the end user is able to use the technology effectively</li> </ul>   |
| • Model and infrastructure design                        | <ul style="list-style-type: none"> <li>Design machine learning model and accompanying infrastructure to ensure that the technology can effectively be integrated into clinical workflows</li> </ul>   | <ul style="list-style-type: none"> <li>Identify a set of input features used by the model to address the problem, making sure to incorporate clinical domain expertise and prior literature</li> <li>Design infrastructure to support clinical decisions in a timely, actionable manner</li> <li>Identify performance metrics and goals that are most important and relevant to stakeholders and end-users</li> </ul> |
| • Clinical workflow application development              | <ul style="list-style-type: none"> <li>Develop the clinical workflow application and integrations with other technologies</li> </ul>  | <ul style="list-style-type: none"> <li>Develop user interface and user experience</li> <li>Integrate with electronic health record to access the required data at the required latency</li> <li>Prototype workflow application with end users</li> </ul>  |
| • Model and infrastructure development                   | <ul style="list-style-type: none"> <li>Develop the machine learning model and infrastructure required to implement model, including integrations with other technologies</li> </ul>   | <ul style="list-style-type: none"> <li>Develop and validate the machine learning model on retrospective data</li> <li>Validate the machine learning model and infrastructure on prospective <i>silent period</i> launch</li> </ul>  |
| • Implementation, change management, and governance      | <ul style="list-style-type: none"> <li>Implement the machine learning model with accompanying education, communication, and governance to ensure accountability and successful adoption</li> </ul>  | <ul style="list-style-type: none"> <li>Establish a governance committee with agreed-upon tasks mission</li> <li>Develop training material to ensure end users effectively use the new technology</li> <li>Communicate broadly about the technology implementation and roles and responsibilities</li> </ul>   |
| • Evaluation plan and partnerships                       | <ul style="list-style-type: none"> <li>Prespecify evaluation plan, target goals, and safety and efficacy monitoring</li> </ul>  | <ul style="list-style-type: none"> <li>Develop internal and external partnerships to ensure rigorous evaluation</li> <li>Register clinical trial</li> </ul>   |

Tabel 1: Langkah Implementasi Machine Learning pada Sepsis Watch



Gambar 1: Swimlane Diagram Sepsis Watch



Gambar 2: Design User Interface Sepsis Watch

Implementasi, Manajemen Perubahan, dan Tata Kelola

Periode diam selama 3 bulan diterapkan sebelum peluncuran, saat Sepsis Watch pertama kali berinteraksi dengan data waktu nyata. Putaran terakhir pemetaan data dilakukan dengan validasi klinis untuk merekonsiliasi perubahan dalam format data, diikuti dengan pengujian model, jalur data, antarmuka pengguna, dan alur kerja dari ujung ke ujung. Versi pertama dari model tersebut mengimplementasikan RNN tanpa MGP dengan pengurangan kinerja yang minimal, yang mencerminkan pertukaran praktis yang sering dibuat antara kinerja model dan upaya rekayasa. Ambang ditetapkan untuk mengoptimalkan nilai prediksi positif dan jumlah peringatan. Hingga 4 peringatan berisiko tinggi per jam disepakati sebagai volume ideal untuk satu pengguna perawat RRT. Pemimpin klinis meninjau 50 kasus berisiko tinggi dengan penundaan 72 jam untuk memvalidasi ambang batas. Akun Sepsis Watch dibuat untuk pemimpin klinis guna memvalidasi keluaran model dan menguji alur kerja. Pemimpin klinis diinstruksikan untuk menghubungi tim rawat inap jika pasien yang diamati membutuhkan tindakan segera. Pada hari rata-rata, sekitar 14 pasien memenuhi kriteria sepsis, dan sekitar 7 pasien berisiko tinggi mengalami sepsis.

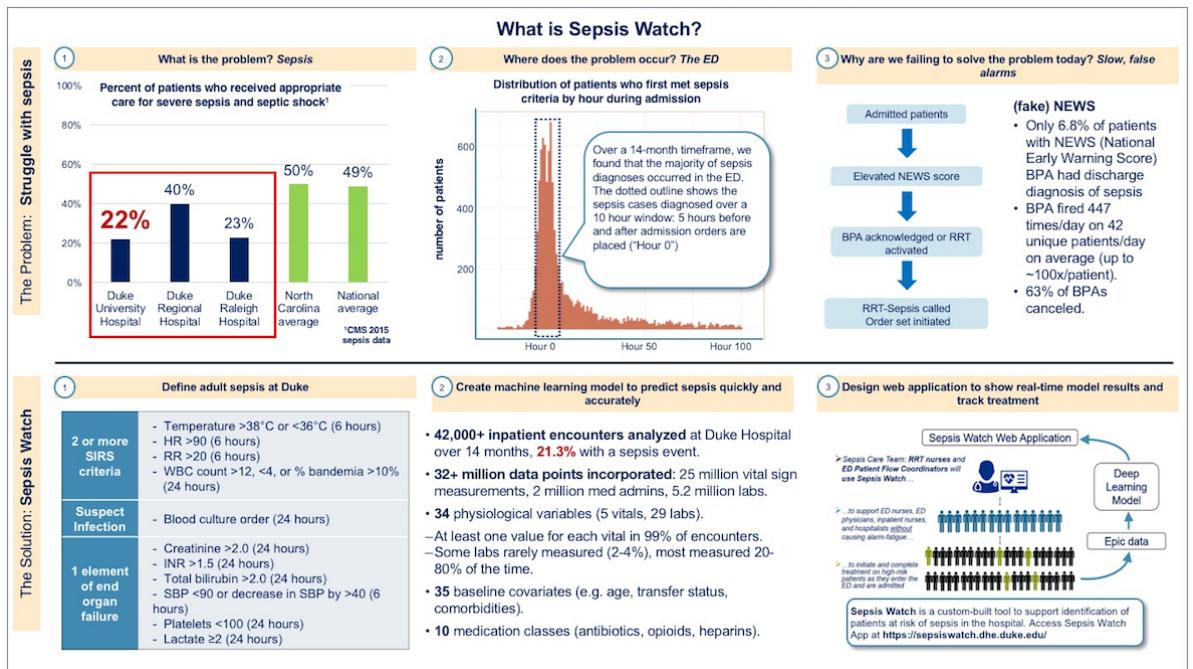
Persiapan Go-Live difokuskan untuk memastikan adopsi dan integrasi Sepsis Watch yang efektif ke dalam perawatan klinis. Sebelum Sepsis Watch, perawat RRT memiliki interaksi minimal dengan dokter UGD. Agar Sepsis Watch memiliki dampak yang diinginkan, 2 peran ini perlu bekerja sama secara erat. Dengan dukungan kepemimpinan senior, titik kontak reguler diprioritaskan untuk menyelaraskan mitra di sekitar visi terpadu tentang alur kerja dan dampak potensial. Dalam 4 minggu menjelang Go-Live, hampir selusin jam per minggu dihabiskan untuk membina hubungan dan saluran komunikasi antar peran. Pertemuan mingguan mempertemukan 1 sampai 2 pemimpin, masing-masing dari perawat RRT, perawat UGD, dokter UGD, dan kelompok pemangku kepentingan rawat inap. Pembaruan akhir minggu dikirim setiap hari Jumat, mencakup kemajuan selama minggu sebelumnya dan tujuan untuk minggu mendatang agar tim tetap selaras. Selama waktu inilah para dokter dan perawat RRT yang terlibat selama 2 tahun proses desain dan pengembangan Sepsis Watch berperan sebagai juara klinis penting yang mempromosikan kepercayaan pada teknologi. Faktanya, ahli statistik dan pengembang utama memiliki interaksi minimal dengan staf garis depan selama 4 minggu menjelang Go-Live. Dokter tanpa peran teknologi informasi formal dalam sistem kesehatan mempromosikan Sepsis Watch di antara rekan-rekan mereka sebagai solusi yang dikembangkan sendiri untuk masalah penting di rumah sakit.

Tujuan berikutnya untuk mendorong adopsi adalah mengomunikasikan visi perubahan secara luas dan memberdayakan tindakan. Tim memulai pelatihan langsung untuk perawat RRT, menekankan kebutuhan mendesak untuk meningkatkan perawatan sepsis di UGD dan peluang untuk meningkatkan hasil dengan Sepsis Watch. Meskipun semua perawat RRT bekerja dalam pengaturan perawatan kritis dan akrab dengan sepsis, pelatihan tentang kriteria diagnostik dan pengobatan untuk sepsis dimasukkan untuk meningkatkan kesadaran dan pemahaman. Tim implementasi menelusuri alur kerja Sepsis Watch dengan perawat RRT secara mendetail menggunakan versi uji aplikasi yang diisi dengan data yang disintesis. Perawat RRT kemudian berinteraksi dengan versi uji aplikasi itu sendiri, mengulangi langkah-langkah alur kerja, dan mengajukan pertanyaan kepada tim implementasi. Pelatihan tatap muka diakhiri dengan diskusi tentang peran dan tanggung jawab serta identifikasi berbagai sumber daya yang tersedia untuk mendukung staf garis depan. Pendidik perawat klinis membantu mengembangkan dan mendistribusikan konten pelatihan di halaman web intranet. Gambar 3 menunjukkan selebaran 1 halaman yang menjelaskan Sepsis Watch. Materi ini juga dikomunikasikan di seluruh unit klinis melalui pertemuan tetap dan layanan daftar email.

Sepsis Watch diluncurkan pada pukul 12 siang Waktu Musim Panas Bagian Timur pada tanggal 5 November 2018. Pengguna pertama adalah perawat RRT yang membantu merancang sistem. Direktur medis UGD memberi tahu dokter UGD untuk mengharapkan panggilan telepon mulai siang hari. Perawat RRT dilengkapi dengan tablet yang dimuat dengan tautan ke aplikasi Sepsis Watch, aplikasi Epic's Canto, beranda pelatihan Sepsis Watch, informasi kontak untuk semua dokter UGD, peta UGD, dan survei 2 menit untuk mengirimkan aplikasi dan umpan balik alur kerja. Cakupan tablet dan Sepsis Watch dibagikan pada akhir setiap shift 12 jam. Sepsis Watch Go-Live berjalan dengan lancar, dan aplikasi tersebut terus digunakan oleh perawat RRT selama uji coba.

Komite tata kelola Sepsis Watch dibentuk untuk memantau efektivitas dan mempromosikan tindakan berbasis luas. Komite termasuk perawat, dokter, dan kepemimpinan administrasi di UGD dan bangsal rawat inap. 4 tujuan utama komite adalah untuk (1) mempromosikan penggunaan aplikasi Sepsis Watch, (2) memberikan pelatihan dan komunikasi yang komprehensif tentang aplikasi dan alur kerja, (3) mengembangkan metode pelaporan untuk melacak volume pasien dan kepatuhan paket, dan (4 ) rencana untuk

keberlanjutan pascapercontohan. Tabel 2 mencantumkan metrik kepatuhan volume dan bundel yang diprioritaskan oleh komite untuk disertakan dalam laporan mingguan. Metrik ini memberikan kejelasan tentang kepatuhan dengan item bundel tertentu, memastikan bahwa volume peringatan masuk akal untuk pengguna perawat RRT tunggal, dan mengidentifikasi kemenangan jangka pendek untuk meningkatkan momentum. Tim implementasi mengirimkan laporan mingguan yang terdiri dari metrik ini ke staf garis depan, termasuk anggota tim perawat RRT, dan komite tata kelola Sepsis Watch.

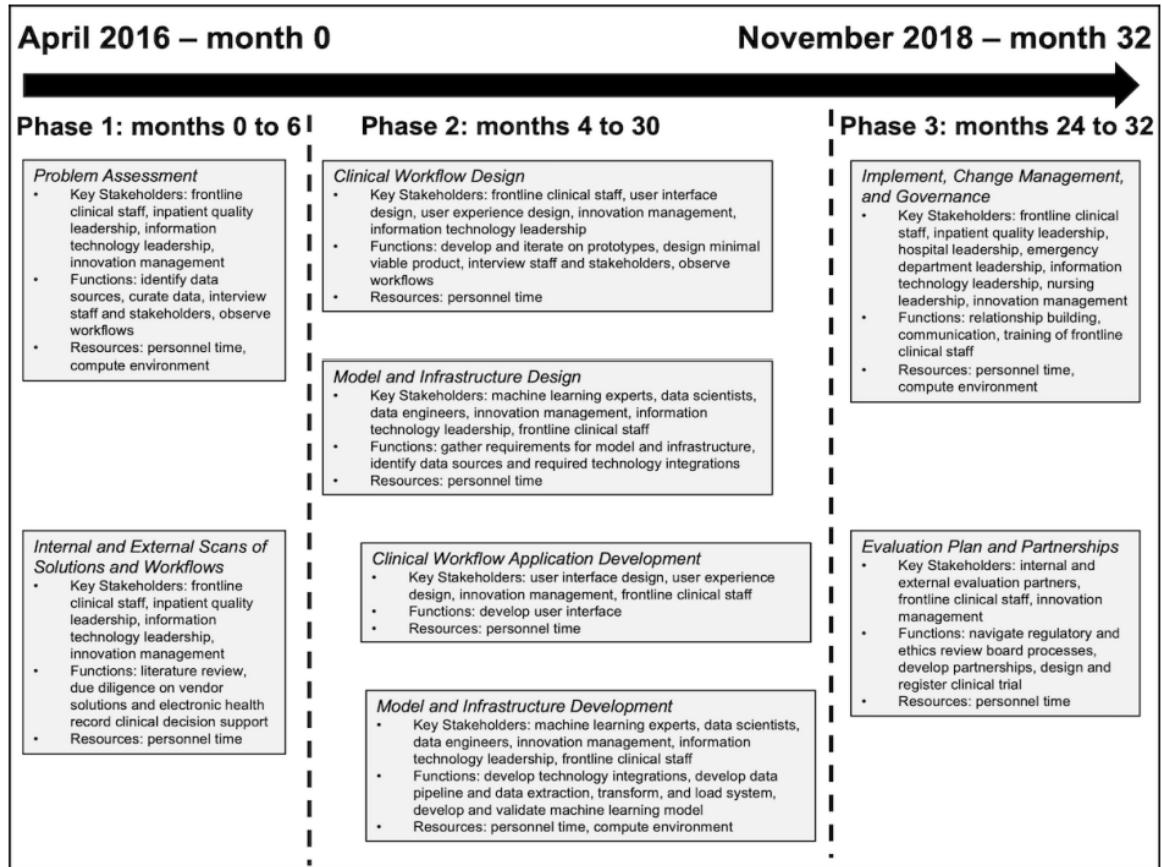


Gambar 3: Sepsis Watch training one-page overview.

| Metric types <sup>a</sup> | Metrics  |
|---------------------------|--|
| Volume                    | Average number of new patients appearing on the Sepsis Watch Triage tab per day  |
| Volume                    | Distribution of new patients appearing on the Sepsis Watch Triage tab, by hour of the day  |
| Volume                    | Median length of time patient remained on the Sepsis Watch Triage tab before being moved to another tab  |
| Volume                    | Average number of patients moved to the Sepsis Bundle Treatment tab per day  |
| Bundle compliance         | 3-hour bundle compliance for patients moved to the Sepsis Watch Treatment tab (comprised of antibiotics, lactate, and blood culture 3-hour bundle components). Includes week-by-week performance |
| Bundle compliance         | Antibiotics administration 3-hour compliance for patients moved to the Sepsis Watch Treatment tab. Includes week-by-week performance   |
| Bundle compliance         | Serum lactate collected 3-hour compliance for patients moved to the Sepsis Watch Treatment tab. Includes week-by-week performance  |
| Bundle compliance         | Blood culture collected 3-hour compliance for patients moved to the Sepsis Watch Treatment tab. Includes week-by-week performance  |

<sup>a</sup>Metrics were chosen by the Sepsis Watch governance committee to present data for 2 distinct patient cohorts: (1) patients who met Sepsis Watch sepsis criteria and (2) patients who were at high risk for meeting Sepsis Watch sepsis criteria as identified by the model.

Tabel 2: Metrik laporan mingguan tata kelola Sepsis Watch.



Gambar 4: Garis waktu langkah-langkah yang terlibat dalam terjemahan Sepsis Watch dari identifikasi masalah hingga integrasi ke dalam perawatan klinis rutin.

## Kesimpulan

Terlepas dari keterbatasannya, keberhasilan integrasi Sepsis Watch ke dalam perawatan klinis rutin menandakan perjalanan melintasi jurang bagi Duke Health. Awalnya, sejumlah kecil dokter dan administrator visioner sangat ingin menggunakan teknologi yang muncul untuk mengatasi masalah klinis yang penting. Seiring perkembangan proyek selama dua tahun, kelompok pemangku kepentingan yang lebih luas menjadi sadar akan dampak potensial dari mengintegrasikan pembelajaran mesin ke dalam perawatan klinis. Permintaan baru untuk aplikasi diumumkan sebulan sebelum Sepsis Watch diluncurkan pada November 2018, dan Duke Institute for Health Innovation menerima rekor jumlah proposal pembelajaran mesin, di mana lima proposal pembelajaran mesin akhirnya dipilih oleh kepemimpinan senior dan diluncurkan pada bulan April 2019. Pada Juni 2019, Sepsis Watch disosialisasikan ke UGD di dua rumah sakit komunitas Duke Health. Banyak tantangan yang dihadapi selama proses integrasi, tetapi fokus pada peningkatan perawatan pasien

memindahkan Sepsis Watch dari konsep ke desain ke produksi. Tidak ada pedoman tentang cara mengintegrasikan pembelajaran mesin ke dalam perawatan klinis, dan masih banyak lagi implementasi yang berhasil diperlukan untuk mengembangkan praktik terbaik. Pembelajaran dari integrasi Sepsis Watch telah menginformasikan proses yang dirancang untuk meningkatkan pelaksanaan proyek pembelajaran mesin dalam sistem kesehatan kita. Pembelajaran ini dapat memberikan arahan kepada tim yang mengejar integrasi pembelajaran mesin ke dalam perawatan di tempat lain.

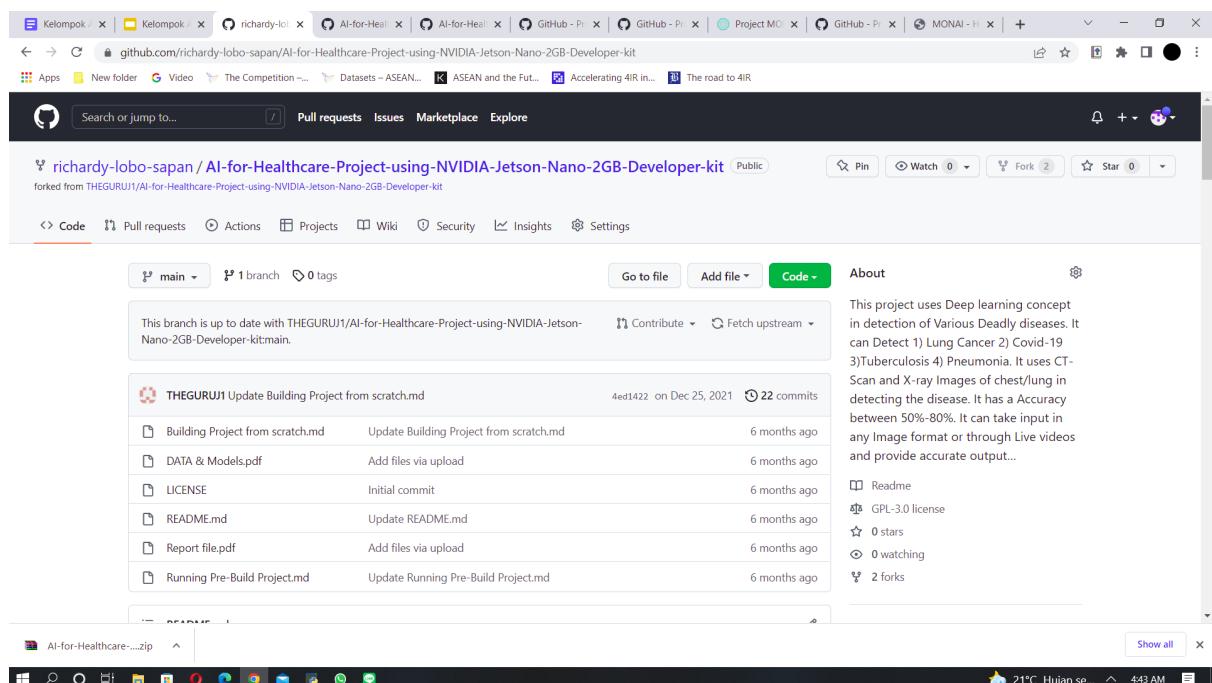
## 2.3 Contoh Implementasi GPU untuk aplikasi Deep Learning: AI-for-Healthcare-Project-using-NVIDIA-Jetson-Nano-2GB-Developer-kit

### Link

### Github:

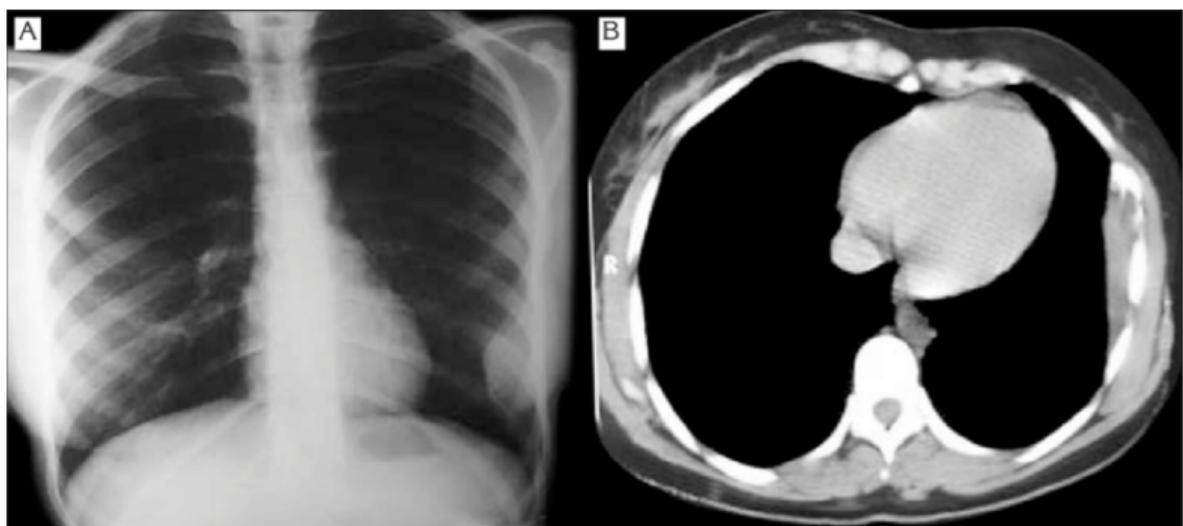
<https://github.com/richardy-lobo-sapan/AI-for-Healthcare-Project-using-NVIDIA-Jetson-Nano-2GB-Developer-kit>

Proyek ini menggunakan konsep Deep learning dalam mendeteksi Berbagai Penyakit Mematikan. Dapat Mendeteksi 1) Kanker Paru 2) Covid-19 3) Tuberkulosis 4) Pneumonia. Menggunakan CT-Scan dan X-ray Gambar dada/paru-paru dalam mendeteksi penyakit. Ini memiliki Akurasi antara 50% -80%. Itu dapat mengambil input dalam format Gambar apa pun atau melalui video Langsung dan memberikan output yang akurat.



Proyek ini menggunakan konsep Deep learning dalam mendeteksi Berbagai Penyakit Mematikan.

- Dapat Mendeteksi 1) Kanker Paru 2) Covid-19 3) Tuberkulosis 4) Pneumonia.
- Menggunakan CT-Scan dan X-ray Gambar dada/paru-paru dalam mendeteksi penyakit.
- Model ini memiliki Akurasi antara 50% -80%.
- Model ini dapat mengambil input dalam format Gambar apa pun atau melalui video Langsung dan memberikan hasil keluaran yang akurat.



| X-ray Image of chest |

| CT-Scan Image of Chest |

### Atribusi

Menggunakan inferensi Jetson @ dusty-nv (<https://github.com/dusty-nv/jetson-inference>)

Menggunakan Datasets:

1. CT-Scan : <https://www.kaggle.com/mohamedhanyyy/chest-ctscan-images/download>
2. Rontgen :  
<https://www.kaggle.com/jtiptj/chest-xray-pneumoniacovid19tuberculosis/download>

### Accessories & Resources

- Jetson Developer Kit (2gb kit)

- Type C power (5V) supply
- Ethernet cable
- HDMI Cable
- Monitor with HDMI cable
- Camera (Logitech C270 HD WEBCAM)
- Keyboard & Mouse (wireless)
- Memory card (more than 32 GB)
- Optional: cooling fan, micro-USB cable(for headless mode)
- Jetson-Inference With Docker File: <https://github.com/dusty-nv/jetson-inference>

Datasets:

- CT-Scan:-<https://www.kaggle.com/mohamedhanyyy/chest-ctscan-images/download>
- X-ray:-<https://www.kaggle.com/jtiptj/chest-xray-pneumoniacovid19tuberculosis/download>

Akurasi Model:





Cara Pembuatan:

Langkah =>

1] Mengumpulkan semua Aksesoris:-

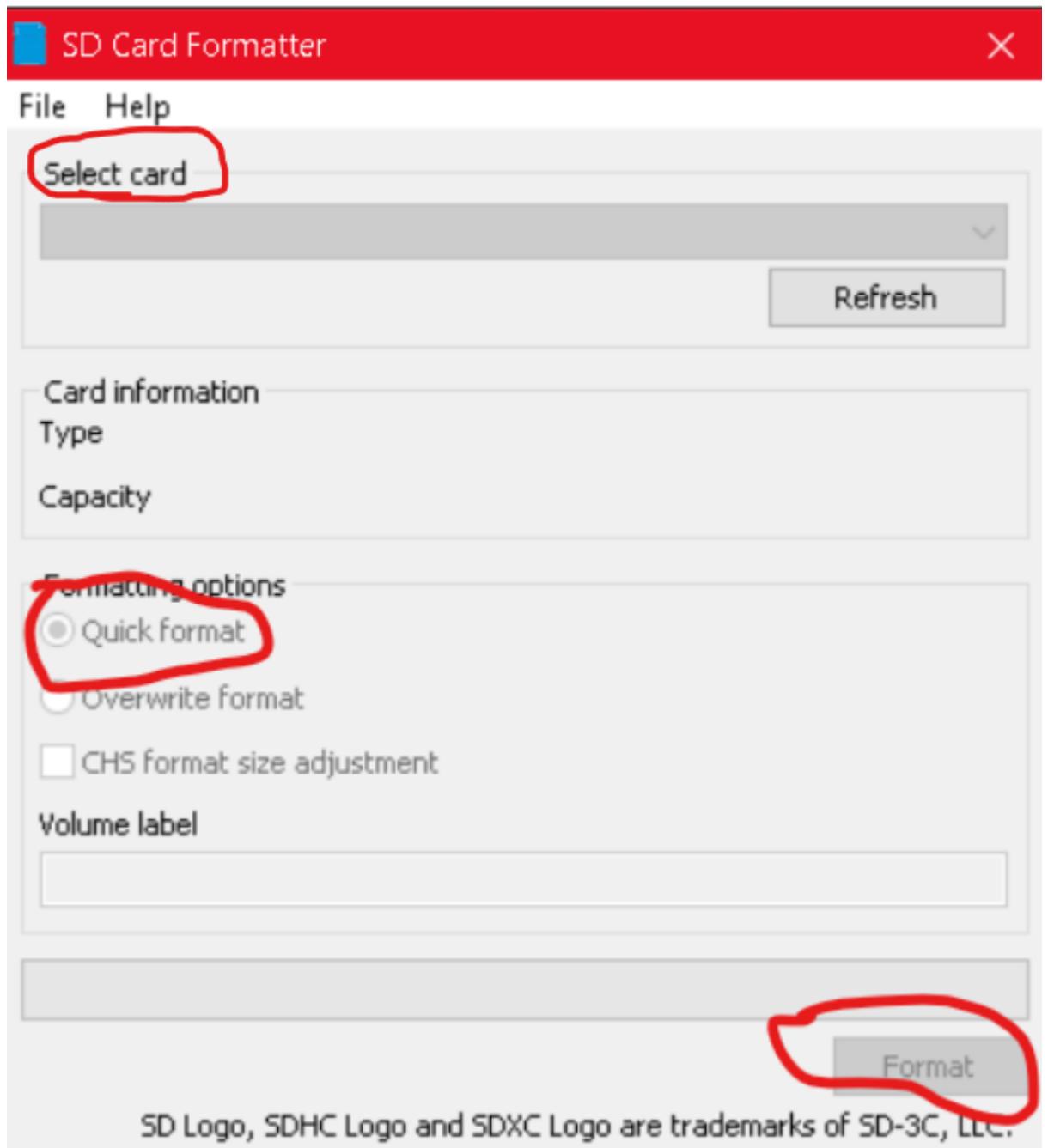
Kita mengumpulkan semua Aksesoris yang disebutkan di atas dari pasar lokal dan mengumpulkan 2GB Developer kit.

2] Mempersiapkan Penyiapan: -

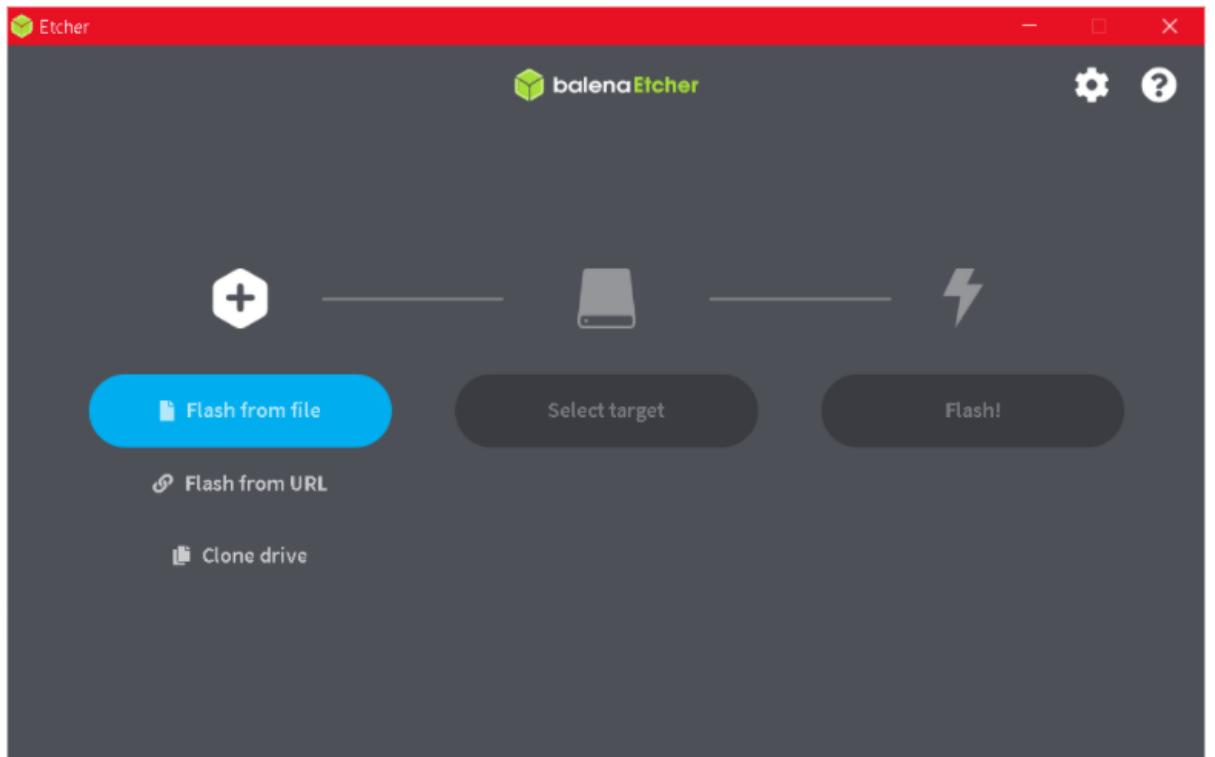
- Hubungkan kartu SD ke PC/Laptop
- Unduh SD Card Image (Untuk kit 2 GB)
- Unduh SD Card Formatter & Instal.



- Quick format the SD card



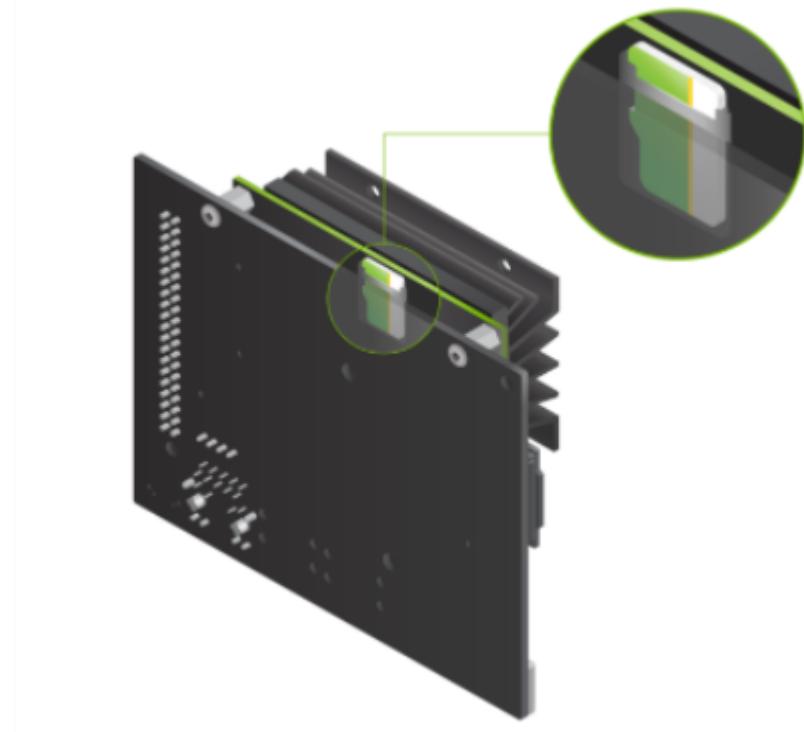
- Download , Install & Launch ETCHER



- pilih downloaded image, pilih perangkat target sebagai kartu memori kemudian flash (membutuhkan lebih dari 10 menit)

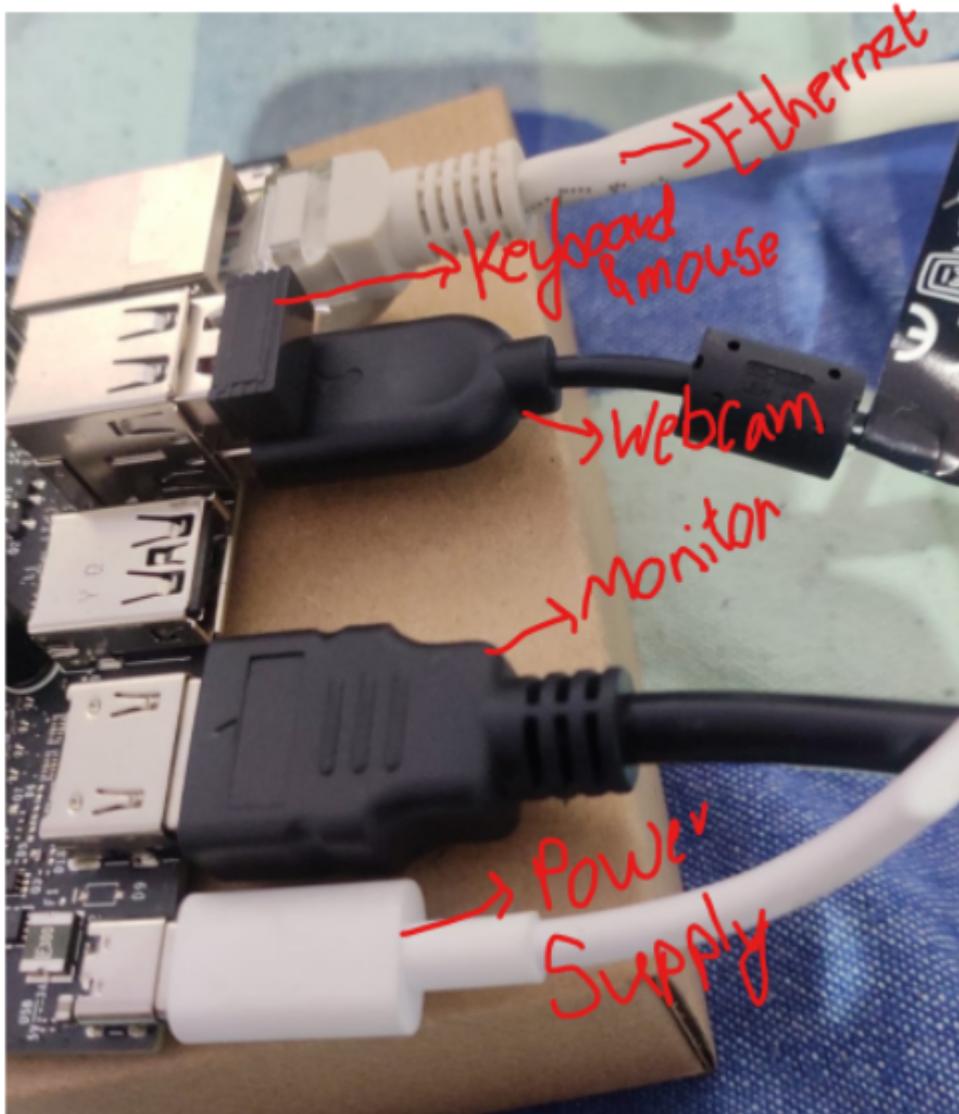
### 3,) Setting up Kit:-

Insert the SD card in kit



Pasang Aksesoris di slot kit seperti yang ditunjukkan di bawah ini





- Nyalakan catu daya & tunggu sistem untuk boot
- Saat Anda boot pertama kali, kit pengembang akan membawa Anda melalui beberapa pengaturan awal, termasuk:
  - Tinjau dan terima perangkat lunak NVIDIA Jetson EULA
  - Pilih bahasa sistem, tata letak keyboard, dan zona waktu
  - Buat nama pengguna, kata sandi, dan nama komputer
  - Konfigurasikan jaringan nirkabel secara opsional
  - Pilih ukuran partisi APP. Disarankan untuk menggunakan ukuran maksimal yang disarankan
  - Buat file swap. Disarankan untuk membuat file swap



#### 4.) Downloading Jetson Inference with Docker Container

Buka Terminal dan ketik perintah berikut

```
git clone --recursive https://github.com/dusty-nv/jetson-inference
```

Tunggu hingga wadah mengunduh (Mungkin membutuhkan waktu 10-15 menit pada koneksi lambat)

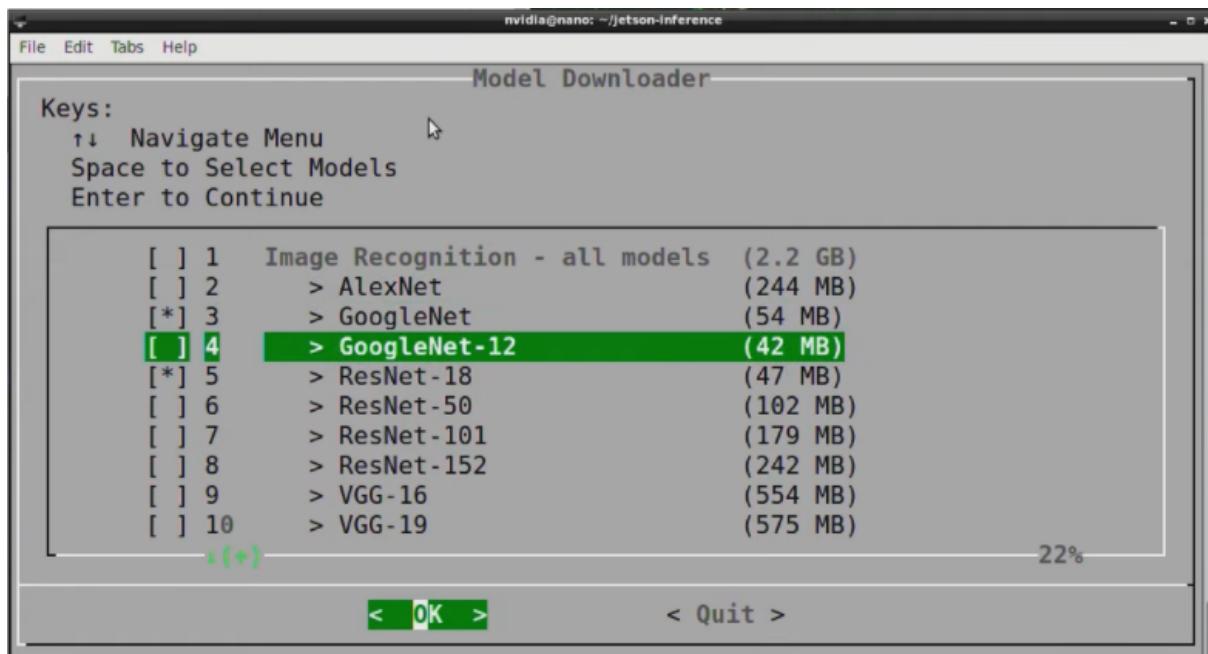
Ubah Direktori ke inferensi jetson menggunakan perintah di bawah ini

```
cd jetson-inferensi
```

Jalankan perintah wadah Docker. Ini akan meminta kata sandi kit Anda. Masukkan kata sandi (di Linux kata sandi yang Anda ketik tidak ditampilkan) dan tekan enter

```
buruh pelabuhan/run.sh
```

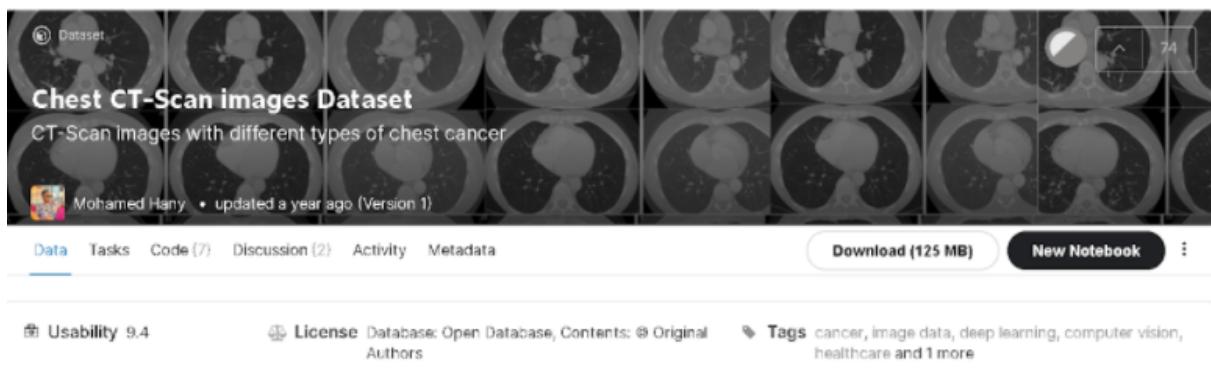
Mungkin perlu waktu dalam Pertama kali menjalankan perintah buruh pelabuhan dan meminta model yang ingin Anda unduh. Unduh model default dan tekan ok



5] Now download the trained dataset from kaggel, we used the following datasets:-

- CT-Scan:-<https://www.kaggle.com/mohamedhanyyy/chest-ctscan-images/download>
- X-ray:<https://www.kaggle.com/jtiptj/chest-xray-pneumoniacovid19tuberculosis/download>
- CT-Scan:-

Ini memiliki Gambar CT-scan paru-paru Pasien Kanker dan Orang Normal



- rontgen:-

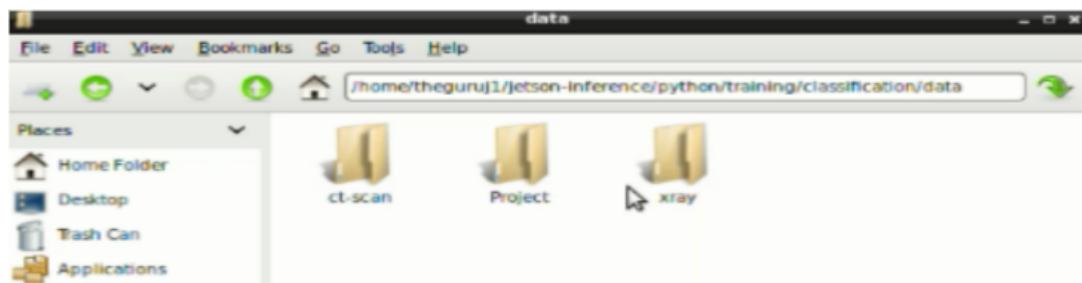


Dataset ini terdiri dari foto rontgen paru pasien Covid-19, pasien Pneumonia, pasien Tuberkulosis dan orang normal.

Sekarang Ekstrak dataset di lokasi:

*Jetson-inference/python/training/classification/data/*

Dengan dataset name. Di folder itu buat file label.txt dan beri nama semua hal yang akan dideteksi oleh kit Anda



6] Melatih Dataset: -

Masukkan perintah berikut

```
python3 train.py --model-dir=models/ct-scan --batch-size=4 --workers=1  
--epochs=35 data/ct-scan
```

Ini akan memakan waktu sekitar 8-10 jam untuk pelatihan dan Kit menjadi panas jadi jangan menyentuhnya.

7] Sekarang ekspor model terlatih dalam file onnx

```
python3 onnx_export.py --model-dir=models/ct-scan
```

8] Lakukan hal yang sama dengan dataset lainnya

9] Sekarang uji proyek: -

```
imagenet      --model=models/ct-scan/resnet18.onnx      --input_blob=input_0  
--output_blob=output_0 --labels=data/ct-scan/labels.txt (lokasi masukan) (Lokasi keluaran)
```

Untuk deteksi berbasis webcam:

```
imagenet      --model=models/ct-scan/resnet18.onnx      --input_blob=input_0  
--output_blob=output_0 --labels=data/ct-scan/labels.txt /dev/video0
```

10] Anda dapat Melatih model sekali lagi untuk meningkatkan akurasi atau Silakan coba dengan model yang berbeda untuk membuat proyek yang luar biasa.

## **BAB III**

## **PENUTUP**

### **1. Kesimpulan**

Dalam karya ilmiah ini, penulis menjelaskan, dengan contoh yang dipilih dengan tepat, bagaimana algoritma pengelompokan Markov bekerja. Penulis menjelaskan tentang (1) Bagaimana konsep dan proses parallel reduction serta cara implementasinya di OpenMP dan GPU Computing (CUDA), (2) Bagaimana contoh makalah tentang teknologi terbaru untuk GPU computing dan Deep learning untuk persoalan terkini bidang data science dan AI (Mis: Masalah Covid-19, big data, telemedicine, blockchain dll.) menggunakan platform terkini (Misal Rapids, KNIME, Torch, PyTorch, Keras, Tensorflow, Clara dll)?, dan (3) Bagaimana Contoh Implementasi GPU untuk aplikasi Deep Learning. Penulis memberikan contoh implementasi yang berhubungan dengan healthcare dan lifesciences. Salah satunya ialah Sepsis Watch. Di sini, penulis memberikan detail tentang bagaimana sistem kesehatan mengintegrasikan teknologi deep learning skala penuh pertama ke dalam perawatan klinis rutin. Sebuah kelompok inovasi menghabiskan lebih dari dua tahun dengan mitra di seluruh organisasi untuk meluncurkan solusi deep learning, Sepsis Watch.

## DAFTAR PUSTAKA

openmp. (2022). Riptutorial.com.

<https://riptutorial.com/openmp/topic/5967/openmp-reductions>

Sendak, M. P., Ratliff, W., Sarro, D., Alderton, E., Futoma, J., Gao, M., Nichols, M., Revoir, M., Yashar, F., Miller, C., Kester, K., Sandhu, S., Corey, K., Brajer, N., Tan, C., Lin, A., Brown, T., Engelbosch, S., Anstrom, K., & Elish, M. C. (2020). Real-World Integration of a Sepsis Deep Learning Technology Into Routine Clinical Care: Implementation Study. *JMIR Medical Informatics*, 8(7), e15182. <https://doi.org/10.2196/15182>