Serial Execution

```
olia@krylov100:~/Diplomatiki/cnn-cifar10_0$ ./cnn-cifar10
CNN for 1200 images
Loading input batch 1...
Load Data time:0.054225 seconds
Load Data to device time:0.133454 seconds
Create Network time:0.040586 seconds
Load Network Parameters time:0.007410 seconds
Create Ouputs time:0.000000 seconds

Net Forward total time:51.396908 seconds
    Time for conv1: 16.491317 seconds
    Time for relu1: 0.108049 seconds
    Time for pool1: 0.216533 seconds
    Time for conv2: 26.625028 seconds
    Time for relu2: 0.024002 seconds
    Time for pool2: 0.048064 seconds
    Time for conv3: 7.835875 seconds
    Time for relu3: 0.000000 seconds
    Time for pool3: 0.011997 seconds
    Time for fc: 0.023997 seconds
    Time for softmax: 0.000000 seconds

  Conv: 50.952220 seconds
  ReLU: 0.132051 seconds
  Pool: 0.276594 seconds
  FC:   0.023997 seconds
  Softmax: 0.000000 seconds

Net Accuracy: 78.25 %
Net Accuracy time:0.000000 seconds
Free memory time:0.000000 seconds
Total time:51.524133 seconds
END!
```

Parallel Softmax only

```
void softmax_forward(float* restrict X, Softmax_Layer* l, float* restrict Y) {

    float max = X[0];
#pragma acc parallel loop present(X,l) copy(max) reduction(max:max)
    for (int i = 1; i < l->out_depth; i++)
        // ...

    float total = 0.0f;
#pragma acc parallel loop present(X,l) copy(total) reduction(+:total)
    for (int i = 0; i < l->out_depth; i++)
```

```
        // ...

    // Normalize and output to sum to one
#pragma acc parallel loop present(X,l,Y) copyin(total)
    for (int i = 0; i < l->out_depth; i++)
        //...
}
```

```
softmax_forward:
    407, Generating present(X[:],l[:])
         Generating copy(max) [if not already present]
         Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        409, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Generating reduction(max:max)
    416, Generating present(X[:],l[:])
         Generating copy(total) [if not already present]
         Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        418, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Generating reduction(+:total)
    418, Generating implicit firstprivate(max)
    422, Generating present(l[:],Y[:],X[:])
         Generating copyin(total) [if not already present]
         Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        426, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

```
Conv: 51.388043 seconds
  ReLU: 0.127992 seconds
  Pool: 0.224374 seconds
  FC:   0.023998 seconds
  Softmax: 0.196176 seconds
```

Ο χρόνος της Softmax αυξήθηκε. Δεν υπολογίζονται οι μεταφορά δεδομένων στον χρόνο αυτό.

Softmax Kernels

```
void softmax_forward(float* restrict X, Softmax_Layer* l, float* restrict Y) {

    float max = X[0];
    float total = 0.0f;
#pragma acc data copyin(max,total) present(l,X,Y)
{
    #pragma acc kernels
    {
    for (int i = 1; i < l->out_depth; i++) //...
    }}}
```

```
softmax_forward:
    407, Generating copyin(total,max) [if not already present]
         Generating present(Y[:],l[:],X[:])
    412, Loop is parallelizable
         Generating NVIDIA GPU code
        412, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Generating implicit reduction(max:max)
    420, Complex loop carried dependence of l->likelihoods->,l->likelihoods
prevents parallelization
         Loop carried dependence of l->likelihoods-> prevents parallelization
         Loop carried backward dependence of l->likelihoods-> prevents
vectorization
         Conditional loop will be executed in scalar mode
         Accelerator serial kernel generated
         Generating NVIDIA GPU code
        420, #pragma acc loop seq
             Generating implicit reduction(+:total)
    427, Loop is parallelizable
         Generating NVIDIA GPU code
        427, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

```
Conv: 51.623373 seconds
  ReLU: 0.128517 seconds
  Pool: 0.332071 seconds
  FC:   0.039997 seconds
  Softmax: 0.091996 seconds
```

Parallel loop, data region

```
void softmax_forward(float* restrict X, Softmax_Layer* l, float* restrict Y) {

    float max = X[0];
    float total = 0.0f;
#pragma acc data copyin(max,total) present(l,X,Y)
{
#pragma acc parallel loop reduction(max:max)
    for (int i = 1; i < l->out_depth; i++)
    // ... Compute max activation

#pragma acc parallel loop reduction(+:total)
    for (int i = 0; i < l->out_depth; i++)
    // ... Compute exponentials and total

#pragma acc parallel loop
    for (int i = 0; i < l->out_depth; i++)
    //...Normalize and output to sum to one
```

```
} // acc data
}
```

```
softmax_forward:
    407, Generating copyin(total,max) [if not already present]
         Generating present(l[:],Y[:],X[:])
         Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        412, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Generating reduction(max:max)
    416, Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        420, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
             Generating reduction(+:total)
    424, Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        428, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

```
Conv: 52.376681 seconds
  ReLU: 0.116132 seconds
  Pool: 0.316277 seconds
  FC:   0.003999 seconds
  Softmax: 0.088111 seconds
```

Parallel Fully Connected

```
void fc_forward(float* restrict X, FC_Layer* l, float* restrict Y) {
    #pragma acc parallel loop present(X,l,Y)
    for (int i = 0; i < l->out_depth; i++) {
        // Calculate dot product of input and weights
        float sum = 0.0;
        for (int j = 0; j < l->in_neurons; j++) {
            int w_idx = j + i * l->in_neurons; // Weight index
            sum += X[j] * l->weights[w_idx];
        }
        sum += l->bias[i];
        Y[i] = sum;
    }
}
```

```
fc_forward:
    243, Generating present(l[:],Y[:],X[:])
         Generating implicit firstprivate(i)
```

```
          Generating NVIDIA GPU code
        249, #pragma acc loop gang /* blockIdx.x */
        252, #pragma acc loop vector(128) /* threadIdx.x */
             Generating implicit reduction(+:sum)
    252, Loop is parallelizable
```

```
Net Forward total time:50.828765 seconds (Serial 50.625525 seconds)
Conv: 50.316362 seconds
  ReLU: 0.088280 seconds
  Pool: 0.240098 seconds
  FC:   0.056027 seconds
  Softmax: 0.107998 seconds
```

FC Kernels

```
void fc_forward(float* restrict X, FC_Layer* l, float* restrict Y) {
#pragma acc kernels present(l,X,Y)
{
    for (int i = 0; i < l->out_depth; i++) {
        float sum = 0.0;
        for (int j = 0; j < l->in_neurons; j++)
        // ... Calculate dot product of input and weights
        sum += l->bias[i];
        Y[i] = sum;
    }
} // acc kernels
}
```

```
fc_forward:
    247, Generating present(Y[:],l[:],X[:])
    249, Loop is parallelizable
         Generating NVIDIA GPU code
        249, #pragma acc loop gang /* blockIdx.x */
        253, #pragma acc loop vector(128) /* threadIdx.x */
             Generating implicit reduction(+:sum)
    253, Loop is parallelizable
```

```
Net Forward total time:50.959538 seconds
Conv: 50.374998 seconds
  ReLU: 0.112185 seconds
  Pool: 0.268259 seconds
  FC:   0.040045 seconds
  Softmax: 0.140053 seconds
```

Parallel Pool

```
void pool_forward(float* restrict X, Pool_Layer* l, float* restrict Y) {

#pragma acc kernels present(X,l,Y)
{
            // For each output feature map
#pragma acc loop independent
    for (int m = 0; m < l->out_depth; m++) {
    #pragma acc loop collapse(2) independent
        for (int j = 0; j < l->out_height; j++) {
            for (int i = 0; i < l->out_width; i++) {
                // Find Max in pooling filter
                float max = -INFINITY;
            #pragma acc loop collapse(2) seq
                for (int p_j = 0; p_j < l->pool_width; p_j++) {
                    for (int p_i = 0; p_i < l->pool_width; p_i++) {
                        int x_j = j * l->stride + p_j;
// Input height index, increased by stride
                        int x_i = i * l->stride + p_i;
// Input width index, increased by stride
                        int x_idx = x_i + (x_j + m * l->in_height) * l->in_width;
// Input index
                        // If in range of input
                        if (x_i >= 0 && x_j >= 0 && x_i < l->in_width && x_j < l-
>in_height) {
                            if (X[x_idx] > max) {
                                max = X[x_idx];
                            } // if max
                        } // if in range
                    } // for p_i
                } // for p_j
                int y_idx = i + l->out_width * (j + m * l->out_height); // Output
index
                Y[y_idx] = max;
            } // for i
        } // for j
    } // for m
} // acc kernels

}
```

```
pool_forward:
    171, Generating present(Y[:],l[:],X[:])
    174, Loop is parallelizable
         Generating NVIDIA GPU code
        174, #pragma acc loop gang /* blockIdx.x */
        176, #pragma acc loop vector(128) collapse(2) /* threadIdx.x */
        177,   /* threadIdx.x collapsed */
        181, #pragma acc loop seq collapse(2)
```

```
          182,   collapsed */
     176, Loop is parallelizable
     177, Loop is parallelizable
```

```
Net Forward total time:51.494460 seconds
  Conv: 50.916264 seconds
  ReLU: 0.140133 seconds
  Pool: 0.120121 seconds
  FC:   0.031997 seconds
  Softmax: 0.157902 seconds
```

Ο χρόνος του pool μειώθηκε, ο συνολικός χρόνος περιλαμβάνει και μεταφορές δεδομένων μεταξύ των επιπέδων που δεν έχουν παραλληλοποιηθεί

Parallel ReLU

```
void relu_forward(float* restrict X, ReLU_Layer* l, float* restrict Y) {

#pragma acc parallel loop present(X,l,Y)
    for (int i = 0; i < l->out_size; i++) {
        Y[i] = (X[i] < 0.0f) ? 0.0f : X[i];
    }

}
```

```
relu_forward:
    119, Generating present(Y[:],l[:],X[:])
         Generating implicit firstprivate(i)
         Generating NVIDIA GPU code
        122, #pragma acc loop gang, vector(128) /* blockIdx.x threadIdx.x */
```

```
Net Forward total time:51.688771 seconds
Conv: 51.191686 seconds
  ReLU: 0.084158 seconds
  Pool: 0.112071 seconds
  FC:   0.044124 seconds
  Softmax: 0.132122 seconds
```

Time for all image dataset

Parallel all layers except convolutional

```
olia@krylov100:~/Diplomatiki/cnn-cifar10_0$ ./cnn-cifar10
CNN for 50000 images
Loading input batch 1...
Loading input batch 2...
Loading input batch 3...
Loading input batch 4...
Loading input batch 5...
Load Data time:0.683571 seconds
Load Data to device time:0.281417 seconds
Create Network time:0.039620 seconds
Load Network Parameters time:0.008463 seconds
Create Ouputs time:0.015042 seconds

Net Forward total time:376.260260 seconds
    Time for conv1: 120.963449 seconds
    Time for relu1: 0.680667 seconds
    Time for pool1: 0.662796 seconds
    Time for conv2: 188.925111 seconds
    Time for relu2: 0.668978 seconds
    Time for pool2: 0.605115 seconds
    Time for conv3: 55.106175 seconds
    Time for relu3: 0.663722 seconds
    Time for pool3: 0.598970 seconds
    Time for fc: 0.645718 seconds
    Time for softmax: 2.492726 seconds

  Conv: 364.994735 seconds
  ReLU: 2.013367 seconds
  Pool: 1.866881 seconds
  FC:   0.645718 seconds
  Softmax: 2.492726 seconds

Net Accuracy: 78.84 %
Net Accuracy time:0.000790 seconds
Free memory time:0.041621 seconds
Total time:375.963642 seconds
END!
```

Serial Execution for 50.000 images

```
olia@krylov100:~/Diplomatiki/cnn-cifar10_0/serial_code$ ./cnn-cifar10
Serial Code
CNN for 50000 images
Loading input batch 1...
Loading input batch 2...
Loading input batch 3...
Loading input batch 4...
Loading input batch 5...
Load Data time:0.916583 seconds
Create Network time:0.000011 seconds
```

```
Load Network Parameters time:0.008689 seconds
Create Ouputs time:0.000453 seconds

Net Forward total time:1440.203875 seconds
    Time for conv1: 462.387537 seconds
    Time for relu1: 4.124741 seconds
    Time for pool1: 7.408484 seconds
    Time for conv2: 743.993717 seconds
    Time for relu2: 1.196957 seconds
    Time for pool2: 2.267751 seconds
    Time for conv3: 216.692938 seconds
    Time for relu3: 0.345922 seconds
    Time for pool3: 0.636112 seconds
    Time for fc: 0.542407 seconds
    Time for softmax: 0.061857 seconds

  Conv: 1423.074192 seconds
  ReLU: 5.667620 seconds
  Pool: 10.312347 seconds
  FC:   0.542407 seconds
  Softmax: 0.061857 seconds

Net Accuracy: 78.84 %
Net Accuracy time:0.003177 seconds
Free memory time:0.055915 seconds
Total time:1441.188703 seconds
END!
```