When Polyhedral Optimizations Meet Deep Learning Kernels

Hrishikesh Vaidya, Akilesh B, Abhishek A Patwardhan, Dr. Ramakrishna Upadrasta

Department of Computer Science and Engineering, IIT Hyderabad, India

Introduction

Deep Neural Networks (DNNs)

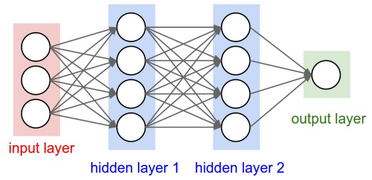

Figure: A 3-layer neural network (Source: cs231n

- Biologically inspired from interactions of neurons in the human brain.
- Widely used in Natural Language Processing, Computer vision, Machine Translation and Bioinformatics tasks.
- Compute and memory intensive programs.
- Challenge: Manually optimizing kernels on parallel heterogeneous architectures.

Motivation

- Automatic parallelization to achieve performance portability.
- DNNs are loop intensive programs.
- DNN computations are similar to well-known BLAS/HPC kernels which benefit by auto-parallelization.

Deep Neural Network layers	BLAS / HPC kernels
Convolutional layer	Stencils, tensor multiplication
Recurrent layer	Iterated Matrix-vector product
LSTM	Sequence of Matrix vector products
Max, Sum Pooling	Max/Sum reductions

Table: Correspondance among DNN layers and HPC kernels

Polyhedral compilation

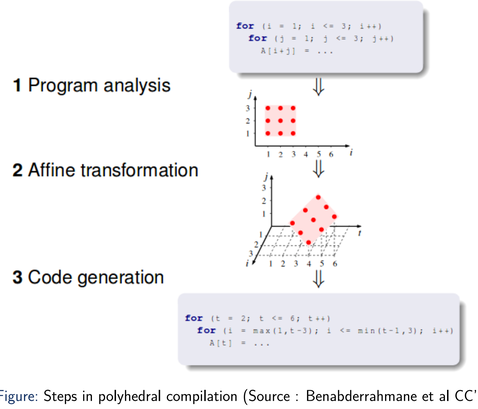

Figure: Steps in polyhedral compilation (Source : Benabderrahmane et al CC'10)

PolyBench/NN: DNN kernels in PolyBench suite

CNN

```
for (n = 0; n < _PB_NN; n++)
  for (k = 0; k < _PB_NK; k++)
    for (p = 0; p < _PB_NP; p++)
      for (q = 0; q < _PB_NQ; q++)
        for (c = 0; c < _PB_NC; c++)
          for (r = 0; r < _PB_NR; r++)
            for (s = 0; s < _PB_NS; s++)
              out_F[n][k][p][q] += W[k][c][r][s] *
                 inp_F[n][c][u*p+NR-r-1][u*q+NS-s-1];
```

Figure: C Code: CNN forward pass

```
for(n = 0; n < _PB_NN; n++)
  for (c = 0; c < _PB_NC; c++)
    for (h = 0; h < _PB_NH; h++)
      for (w = 0; w < _PB_NW; w++)
        for (k = 0; k < _PB_NK; k++)
          for (r = 0; r < _PB_NR; r++)
            for (s = 0; s < _PB_NS; s++)
              for (p = 0; p < _PB_NP; p++)
                for (q = 0; q < _PB_NQ; q++)
                  if((u*p - (h - NR + r + 1) ==0) && (u*q - (w - NS + s + 1) ==0))
                    err_in[n][c][h][w] += W[k][c][r][s] * err_out[n][k][p][q];
```

Figure: C Code: CNN backward pass

Table: Program Parameters for CNN

N	Number of Input Images in batch
C	Number of Input feature maps
K	Number of Output feature maps
P × Q	Size of output feature map
R × S	Size of filter kernel
U,V	Stride parameters

RNN

```
for (t = 0; t < _PB_NT; t++)
{
    for(s1 = 0; s1 < _PB_NS; s1++)
    {
        for(p = 0; p < _PB_NP; p++)
            s_F[t][s1] += U[s1][p] * inp_F[t][p];

        if(t > 0)
            for(s2 = 0; s2 < _PB_NS; s2++)
                s_F[t][s1] += W[s1][s2] * s_F[t-1][s2];
    }
    for(q = 0; q < _PB_NQ; q++)
        for(s1 = 0; s1 < _PB_NS; s1++)
            out_F[t][q] += V[q][s1] * s_F[t][s1];
}
```

Figure: C Code: RNN forward pass

```
for (t = _PB_NT - 1; t > 0; t--)
{
    for(q = 0; q < _PB_NQ; q++)
        for(s = 0; s < _PB_NS; s++)
            delV[q][s] += err_out[t][q] * s_F[t][s];

    for(s = 0; s < _PB_NS; s++)
    {
        delTA[s] = 0;
        for(q = 0; q < _PB_NQ; q++)
            delTA[s] += V[q][s] * err_out[t][q];
    }
    for(step=t+1; step > max(0, t - bptt_trunc); step--)
    {
        if(step > 0)
            for(r = 0; r < _PB_NS; r++)
                for(s = 0; s < _PB_NS; s++)
                    delW[r][s] += delTA[r] * s_F[step - 1][s];

        for(s = 0; s < _PB_NS; s++)
            for(p = 0; p < _PB_NP; p++)
                delU[s][p] += delTA[s] * inp_F[step][p];

        for(r = 0; r < _PB_NS; r++)
        {
            delTB[r] = 0;
            for(s = 0; s < _PB_NS; s++)
                delTB[r] += delTA[s] * W[s][r];
        }

        for(r = 0; r < _PB_NS; r++)
            delTA[r] = delTB[r];
    }
}
```

Figure: C Code: RNN backward pass

PolyBench/NN (cont..)

Table: Program Parameters for RNN

T	Number of time steps
P	Size of input vector
Q	Size of output vector
S	Size of hidden vector
BPTT	Truncated Unroll factor

LSTM

```
for (t = 0; t < _PB_T; t++)
{
    for(s1 = 0; s1 < _PB_S; s1++)
    {
        i[s1] = 0.0;
        for(p = 0; p < _PB_P; p++)
            i[s1] += U_i[s1][p] * inp_F[t][p];

        if(t > 0)
            for(s2 = 0; s2 < _PB_S; s2++)
                i[s1] += W_i[s1][s2] * s_F[t-1][s2];
    }

    for(s1 = 0; s1 < _PB_S; s1++)
    {
        f[s1] = 0.0;
        for(p = 0; p < _PB_P; p++)
            f[s1] += U_f[s1][p] * inp_F[t][p];

        if(t > 0)
            for(s2 = 0; s2 < _PB_S; s2++)
                f[s1] += W_f[s1][s2] * s_F[t-1][s2];

    }
    for(s1 = 0; s1 < _PB_S; s1++)
    {
        o[s1] = 0.0;
        for(p = 0; p < _PB_P; p++)
            o[s1] += U_o[s1][p] * inp_F[t][p];

        if(t > 0)
            for(s2 = 0; s2 < _PB_S; s2++)
                o[s1] += W_o[s1][s2] * s_F[t-1][s2];
    }
    for(s1 = 0; s1 < _PB_S; s1++)
    {
        g[s1] = 0.0;
        for(p = 0; p < _PB_P; p++)
            g[s1] += U_g[s1][p] * inp_F[t][p];

        if(t > 0)
            for(s2 = 0; s2 < _PB_S; s2++)
                g[s1] += W_g[s1][s2] * s_F[t-1][s2];
    }

    if(t > 0)
        for (b = 0; b < ns; b++)
            c_F[t][b] = c_F[t-1][b] * f[b] + g[b] * i[b];

    for (b = 0; b < ns; b++)
        s_F[t][b] = c_F[t][b] * o[b];

}
```

Figure: C Code: LSTM forward pass

Table: Program Parameters for LSTM

T	Number of time steps
P	Size of input vector
Q	Size of output vector
S	Size of hidden vector

Performance evaluation

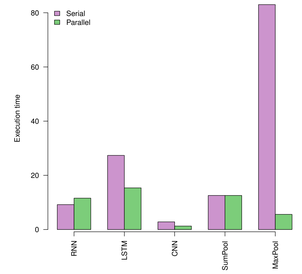

Figure: Execution times for forward pass

Figure: Execution times for backward pass

Conclusions

- DNN kernels are **amenable** for polyhedral optimizations.
- **RNN, LSTM:** Purely affine; **CNN, Pool:** Almost affine.
- Open-source implementation of varieties of DNN kernels as **affine control** loop nests.
- Polyhedral transformations extracts coarse-grain parallelization, improves data-locality.
- Upto $51\times$ speedups by automatic loop transformations via PLUTO polyhedral compiler.

Contact Information

- Web: http://www.iith.ac.in/ ramakrishna/
- Email: {cs13b10[35,42], cs15mtech11015, ramakrishna}@iith.ac.in
- Github: https://github.com/hrishikeshv

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad