Energy Complexity Model for Convolutional Neural Networks

Jiří Šíma

sima@cs.cas.cz

Institute of Computer Science (ICS)
Czech Academy of Sciences, Prague, Czechia

joint work with

Petra Vidnerová

ICS, Czech Academy of Sciences, Prague, Czechia

Vojtěch Mrázek

Faculty of Information Technology
Brno University of Technology, Brno, Czechia

Efficient Processing of Deep Neural Networks (DNNs)

- DNNs are widely used for many artificial intelligence (AI) applications including computer vision, speech recognition, natural language processing, robotics etc.

- DNNs achieve state-of-the-art accuracy on many AI tasks at the cost of high computational complexity (tens of millions of operations for a single inference)

- energy efficiency of DNN implementations in low-power hardware operated on batteries (e.g. cellphones, smartwatches, smart glasses) becomes crucial

$\longrightarrow$ reducing the energy cost of DNNs:

1. approximate computing methods (e.g. low floating-point precision, approximate multipliers) in error-tolerant applications such as image classification

2. hardware design: energy-efficient implementations of DNNs on various hardware platforms including GPUs, FPGAs, in-memory computing architectures

Energy Consumption of DNNs

- the power consumption of a specific DNN hardware implementation can be measured or calculated/estimated (using physical laws)
- a plethora of methods that minimize the energy consumption of a given DNN on various hardware architectures
(Sze,Chen,Yang,Emer:Efficient Processing of Deep Neural Networks,2020)
- automated by software tools, for example, the `Timeloop` program maps a convolutional layer specified by its parameters onto a given hardware architecture (e.g. Simba, Eyeriss) that is optimal in terms of power consumption estimated by `Accelergy` tool which reports the energy statistics
- it has been empirically observed that the energy for DNN inference is mainly consumed by
  1. data movement inside a memory hierarchy (approx. 70%) corresponding to the data energy E_{data}
  2. multiply-and-accumulate (MAC) operations (approx. 30%): $S \leftarrow S + wx$ on floats S, w, x, corresponding to the computation energy E_{comp}

$$\longrightarrow \quad E = E_{\text{data}} + E_{\text{comp}}$$

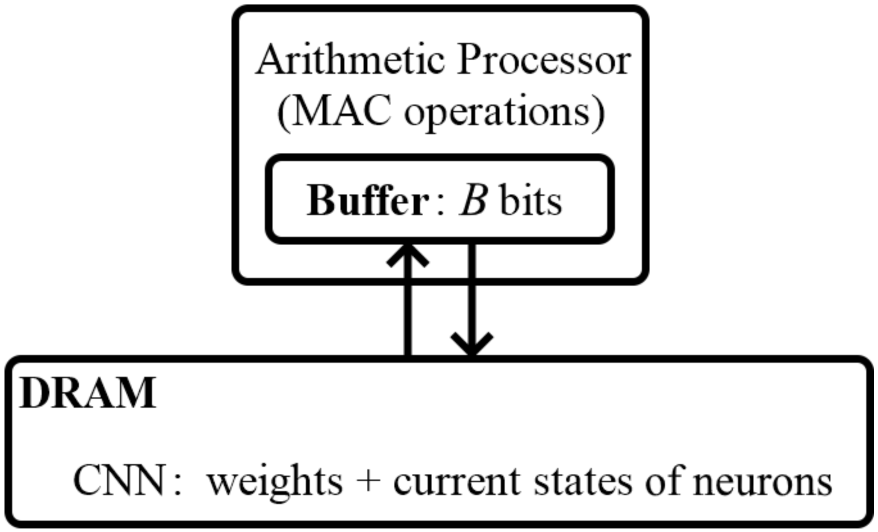
Motivations for Energy Complexity Model of DNNs

- the evaluation of real power consumption for individual DNN implementations varies for different hardware architectures depending on their specific parameters, which prevents from machine-independent exploration of energy complexity
- a formal computational model for defining a robust energy measure for DNNs, quantified asymptotically using Big O notation
  (by analogy to computation time and memory space defined by Turing machines)
- lower bounds on energy complexity can establish principal limits of DNNs

$\longrightarrow$ A Simplified Hardware-Independent Model of Energy Complexity for DNNs:

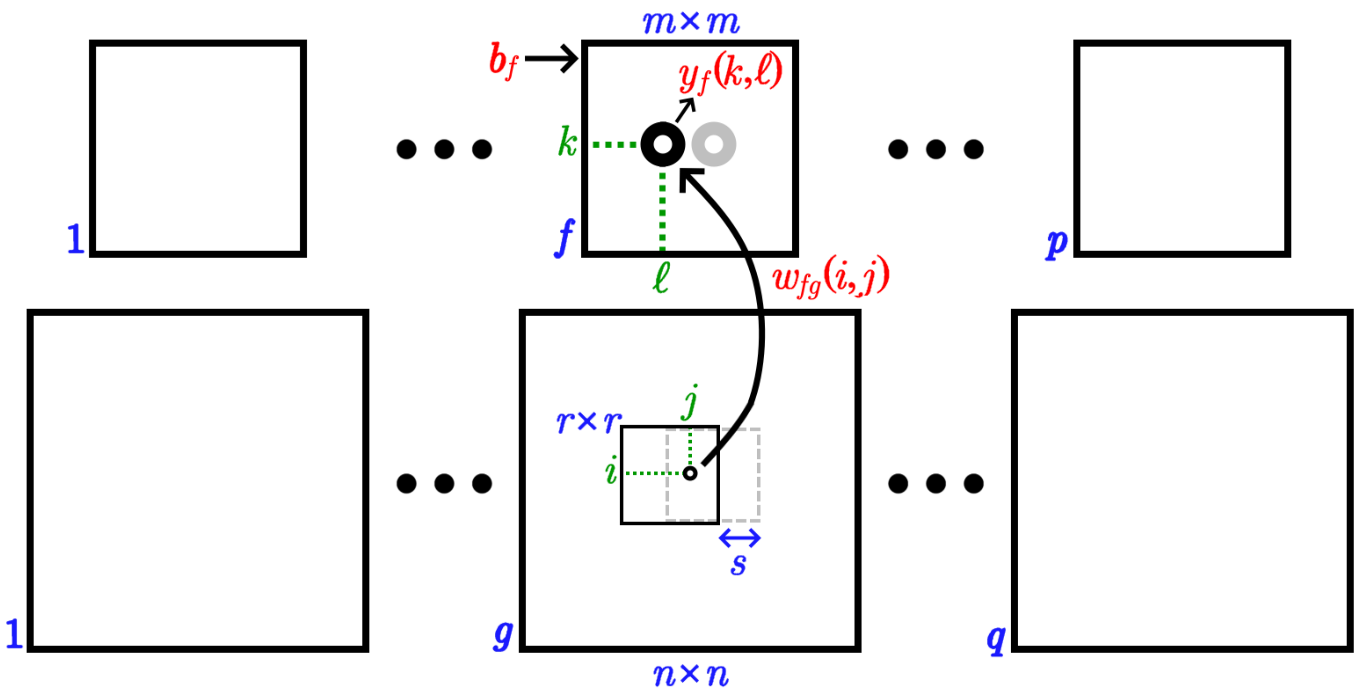
- abstracts from hardware implementation details, ignoring specific aspects and parameters of real-world machine
- preserves the asymptotic energy of DNN inference
- is defined (for simplicity) for a separate layer of a convolutional neural network (CNN), avoiding global energy optimization across multiple CNN layers

Energy Complexity Model for CNNs



- only two memory levels called DRAM (large, slow, and cheap memory) and Buffer of limited capacity B bits (small, fast, and expensive memory)
- CNN weights and states are stored in DRAM
- arithmetic operations are performed over numerical data stored in Buffer
- the dataflow controls the transfer of data between DRAM and Buffer
- the main idea: the three arguments stored in DRAM, input x, weight w, and accumulated output S of each MAC operation $S \leftarrow S + wx$ performed for evaluating a given convolutional layer, must occur in Buffer simultaneously

A Convolutional Layer



p feature maps $f \in \{1, \ldots, p\}$
of size $m \times m$ neurons
(depth $= p$, height=width $= m$)

receptive fields
of size $r \times r$ neurons
with stride s

q input feature maps
$g \in \{1, \ldots, q\}$
of size $n \times n$ neurons
$m \approx \frac{n}{s}$

$y_f(k, \ell)$ is the state of neuron $(k, \ell) \in \{1, \ldots, m\}^2$ in feature map $f \in \{1, \ldots, p\}$

$$y_f(k,\ell) = \mathrm{ReLU}\left(b_f + \sum_{g=1}^{q} \sum_{i=1}^{r} \sum_{j=1}^{r} w_{fg}(i,j) \cdot y_g\big((k-1)s + i\,,\,(\ell-1)s + j\big) \right)$$

where $\mathrm{ReLU}(x) = \max(0, x)$, b_f is the bias of f, and $w_{fg}(i,j)$ is the filter weight of neuron $(i,j) \in \{1, \ldots, r\}^2$ in a receptive field of f over the input feature map $g \in \{1, \ldots, q\}$

$\longrightarrow$ the number of MAC operations (#MACs) is $p\,m^2 \cdot q\,r^2 \;\approx\; p\,q\,n^2\,\frac{r^2}{s^2}$

The Energy Complexity Measure for a Convolutional Layer

$$E = E_{\text{data}} + E_{\text{comp}}$$

for a given dataflow used to evaluate a convolutional layer:

E_{data} is # DRAM accesses $\times$ # bits b in floating-point numbers

$E_{\text{comp}} = C_b \cdot p\,q\,m^2 r^2 \approx p\,q\,n^2\,\frac{r^2}{s^2}$ is proportional to # MACs (on data in Buffer)

where C_b is a non-uniform constant related to a b-bit floating-point MAC circuit

A Simple Lower Bound on the Data Energy

$E_{\text{data}} \;\geq\; b \cdot$ # MACs divided by $\left(\frac{B-1}{2}\right)^2 =$ the maximum number of new triplets (input, output, weight) (i.e. the MAC arguments) that can meet in Buffer of capacity B bits after reading one number into Buffer (i.e. one DRAM access)

$$\longrightarrow \quad E_{\text{data}} = \Omega\left(p\,q\,n^2\,\frac{r^2}{s^2}\right) \quad \text{for constant Buffer capacity } B$$

A Partition of the Input Feature Map

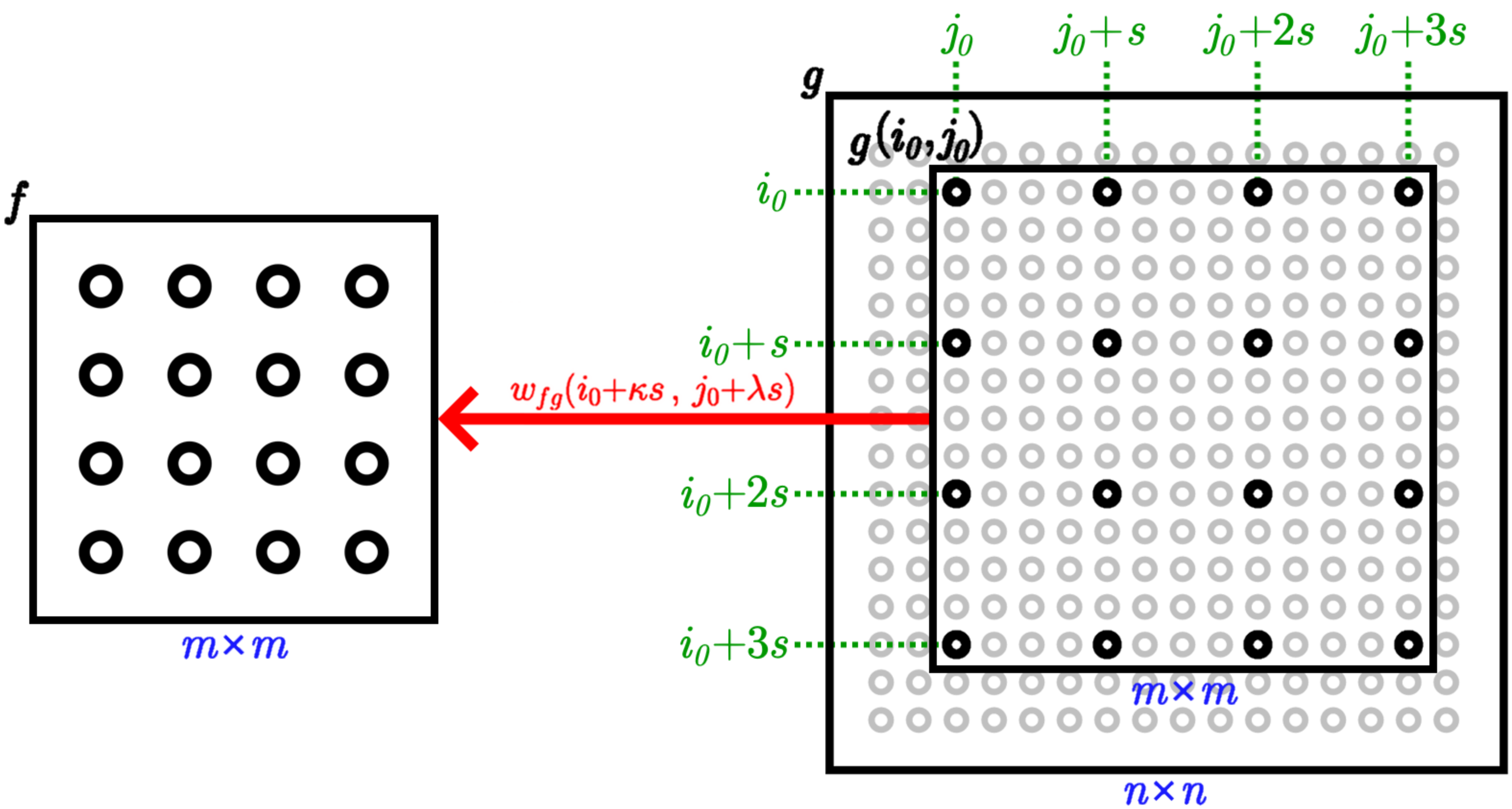
a partition of input feature map $g \in \{1, \ldots, q\}$ of $n \times n$ neurons,

$$g = \bigcup_{i_0, j_0 \in \{1,\ldots,s\}} g(i_0\,, j_0) \quad \text{into } s^2 \text{ grid submaps}$$

$$g(i_0, j_0) = \Big\{ \big((k-1)s + i_0\,,\ (\ell-1)s + j_0 \big) \mid k, \ell \in \{1, \ldots, m\} \Big\} \text{ of } m \times m \text{ neurons}$$

that share the same weights $w_{fg}(i_0 + \kappa s\,,\ j_0 + \lambda s)$ for every (admissible) integer κ, λ

$$\text{in} \quad y_f(k, \ell) = \mathrm{ReLU}\left(b_f + \sum_{g=1}^{q} \sum_{i=1}^{r} \sum_{j=1}^{r} w_{fg}(i, j) \cdot y_g\big((k-1)s + i\,,\ (\ell-1)s + j\big) \right)$$



A Dataflow with Write-Once Outputs (similarly for read-once inputs)

each output is completely evaluated at once in `Buffer` before it is written to `DRAM` while each weights is read into `Buffer` only once:

for all feature maps $f \in \{1, \ldots, p\}$ **do**
  read bias b_f into `Buffer`;
  for all $k, \ell \in \{1, \ldots, m\}$ **do** $S_f(k, l) \leftarrow b_f$ **enddo**;   {initialization of $m \times m$ weighted sums}
  for all input feature maps $g \in \{1, \ldots, q\}$ **do**
    for all $i_0, j_0 \in \{1, \ldots, s\}$ **do**   {for all grid submaps $g(i_0, j_0)$ from the partition of g}
      for all $(k, \ell) \in g(i_0, j_0)$ **do read** $y_g(k, \ell)$ into `Buffer` **enddo**;   {reading $m \times m$ submap inputs}
      for all admissible integer κ, λ **do**   {for all the weights shared by submap $g(i_0, j_0)$}
        $i \leftarrow i_0 + \kappa s$; $j \leftarrow j_0 + \lambda s$; $\{1 \leq i, j \leq r\}$
        read a single weight $w_{fg}(i, j)$ into `Buffer`;
        for all $k, \ell \in \{1, \ldots, m\}$ **do**   {all MACs with the weight $w_{fg}(i, j)$}
          $S_f(k, l) \leftarrow S_f(k, l) + w_{fg}(i, j) \cdot y_g\big((k-1)s + i\,,\,(\ell - 1)s + j\big)$
        enddo
      enddo   {next shared weight}
    enddo   {next submap}
  enddo;   {next input feature map}
  for all $k, \ell \in \{1, \ldots, m\}$ **do write** $y_f(k, l) = \mathrm{ReLU}\big(S_f(k, l)\big)$ to `DRAM` **enddo** {writing $m \times m$ outputs}
enddo   {next feature map}

The Capacity of `Buffer`

the used `Buffer` memory: $B = b \cdot (2m^2 + 1)$

- m^2 accumulated outputs of feature map f
- m^2 inputs from grid submap $g(i_0, j_0)$
- 1 shared weight

$\longrightarrow$ a realistic assumption on the `Buffer` capacity:

$$B \geq b \cdot (2m^2 + 1)$$

e.g. `Buffer` capacities in kilobytes required for convolutional layers in AlexNet:

AlexNet layer	1	2	3	4	5
m	55	27	13	13	13
$2m^2 + 1$	6051	1459	339	339	339
$b = 8$ bits	5.91 kB	1.42 kB	0.33 kB	0.33 kB	0.33 kB
$b = 16$ bits	11.82 kB	2.85 kB	0.66 kB	0.66 kB	0.66 kB
$b = 32$ bits	23.64 kB	5.7 kB	1.32 kB	1.32 kB	1.32 kB

An Upper Bound on Data Energy E_{data}

the data energy E_{data} in terms of #DRAM accesses for inputs, outputs, and weights:

$$E_{\text{data}} = E_{\text{weights}} + E_{\text{outputs}} + E_{\text{inputs}} \qquad \text{where}$$

$$E_{\text{inputs}} = b \cdot p\, q\, n^2 \qquad E_{\text{outputs}} = b \cdot p\, m^2 \approx b \cdot p\, \frac{n^2}{s^2} \qquad E_{\text{weights}} = b \cdot p(q\, r^2 + 1)$$

$\longrightarrow$ an upper bound:

$$E_{\text{data}} \;\leq\; b \cdot p\,(q\, n^2 + m^2 + q\, r^2 + 1) = O\left(p\left(qn^2 + \frac{n^2}{s^2} + qr^2\right)\right)$$

the asymptotic theoretical energy complexity in terms of individual convolutional layer parameters (others are constant):

$E_{\text{data}} = O(p)$ where p is the number of feature maps (i.e. depth)

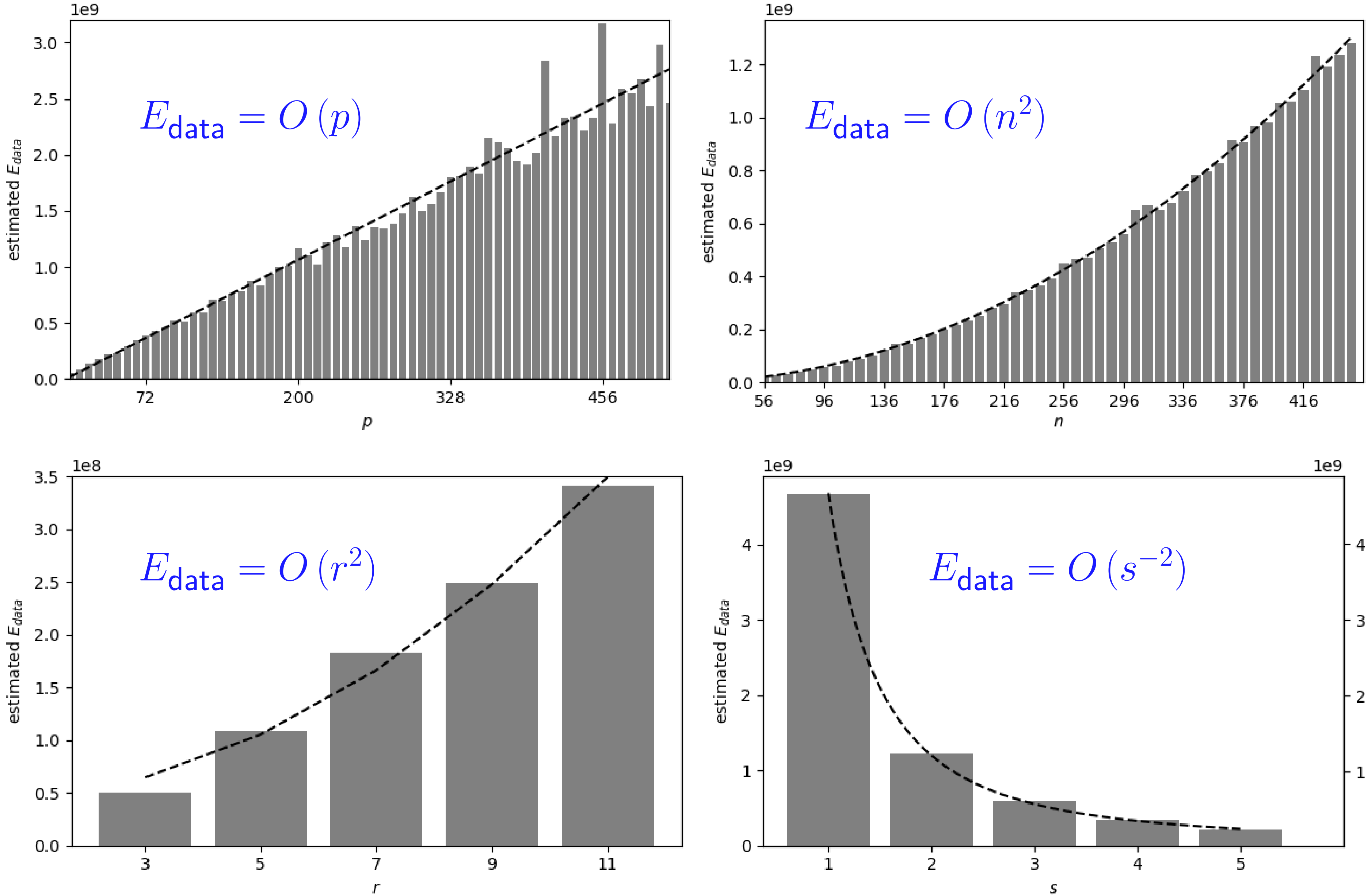
$E_{\text{data}} = O(n^2)$ where n is the the size of input feature maps (i.e. height=width)

$E_{\text{data}} = O(r^2)$ where r is the size of receptive fields

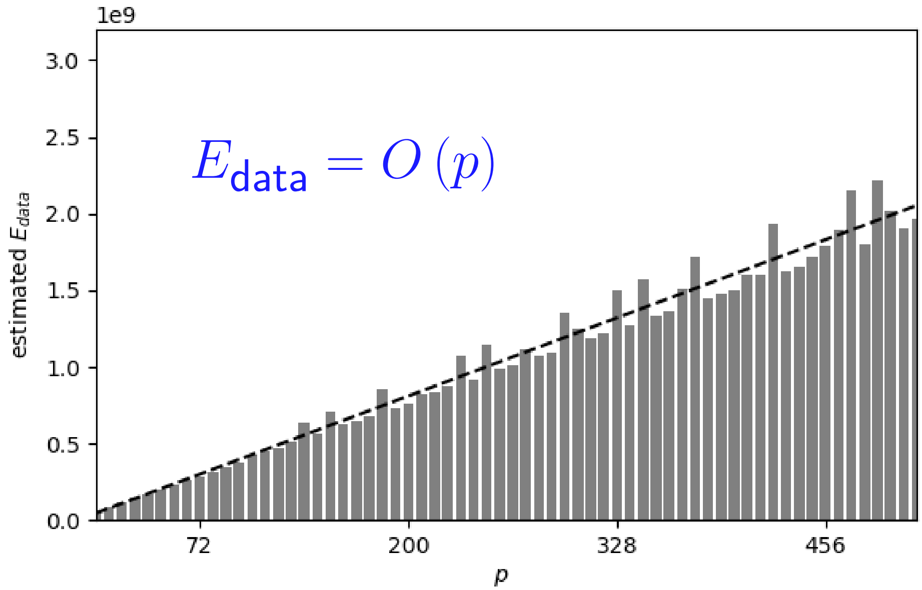
$E_{\text{data}} = O(s^{-2})$ where s is the stride

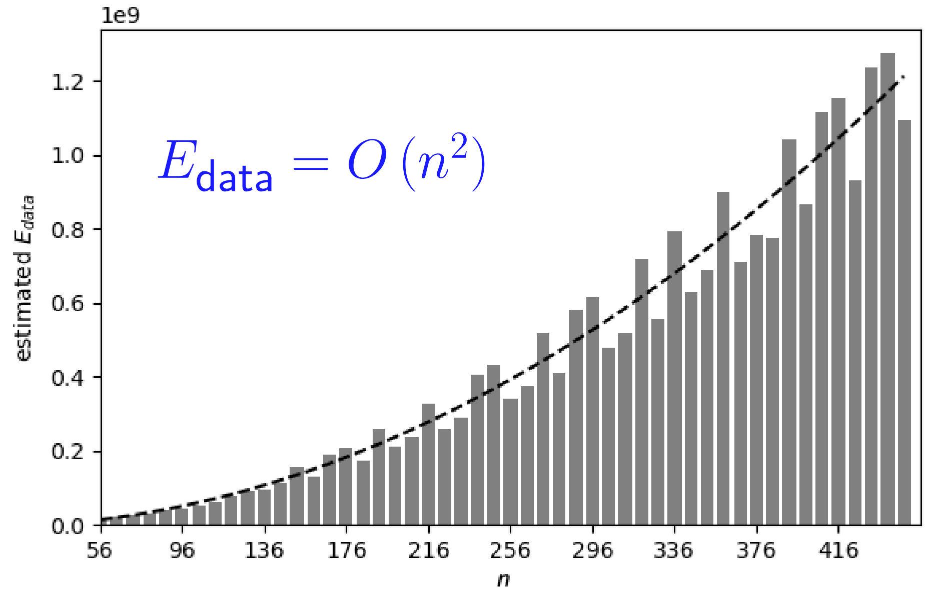
fits very well (by linearity/quadraticity statistical tests) the real power consumptions estimated by the `Timeloop/Accelergy` software platform that maps a convolutional layer of given parameters onto the Simba and Eyeriss hardware architectures:

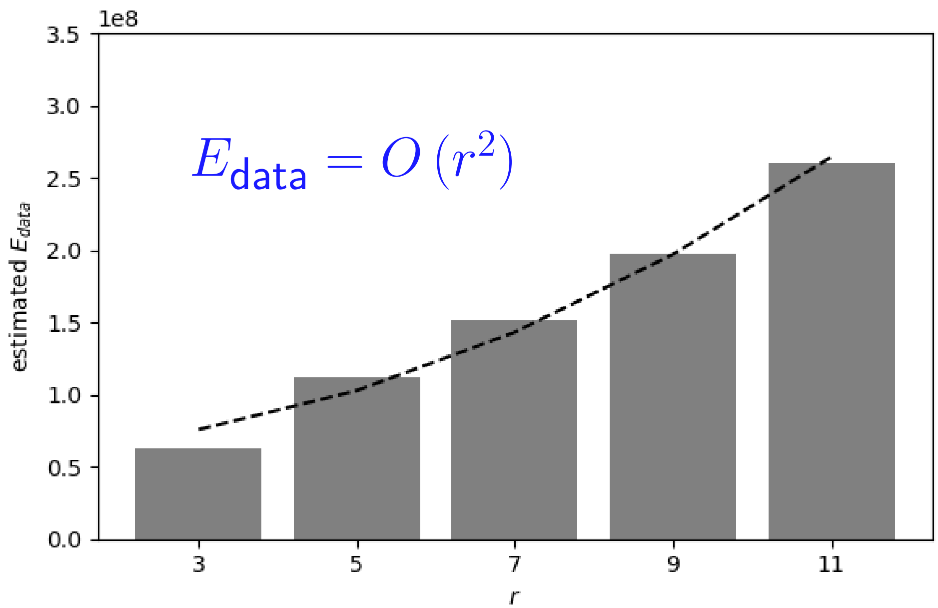
Experimental Validation of Energy Complexity Model for Simba

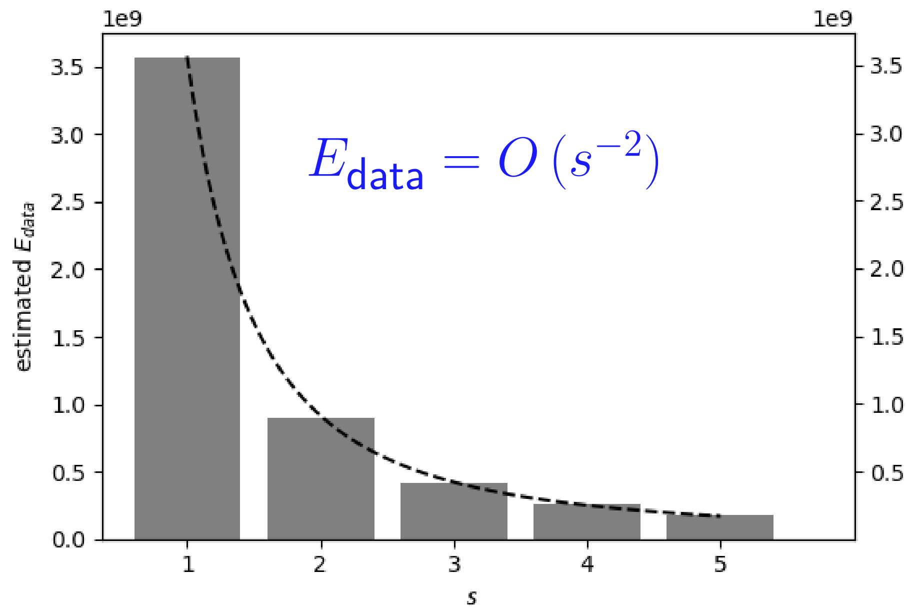


Experimental Validation of Energy Complexity Model for Eyeriss









A Summary

- we have introduced a machine-independent model of energy complexity for CNNs
- in this model, we have proposed a dataflow with write-once outputs (or read-once inputs) and read-once weights for evaluating convolutional layers
- this provides an upper bound on the theoretical energy complexity of CNNs which fits asymptotically very well the power consumption estimates of their various hardware implementations
- we have shown a simple lower bound on energy of convolutional layers which establishes the principal limit on energy efficiency of CNNs

Open Problems

- an experimental validation of the energy complexity model for combined parameters of convolutional layer **?**
- the matching lower bound on energy of convolutional layers for `Buffer` of non-constant size **?**