Fast N^2 t-distributed Stochastic Neighbor Embedding

Andreas Blöchliger
Marc Fischer
Alberto Montes
Marko P. Trauber

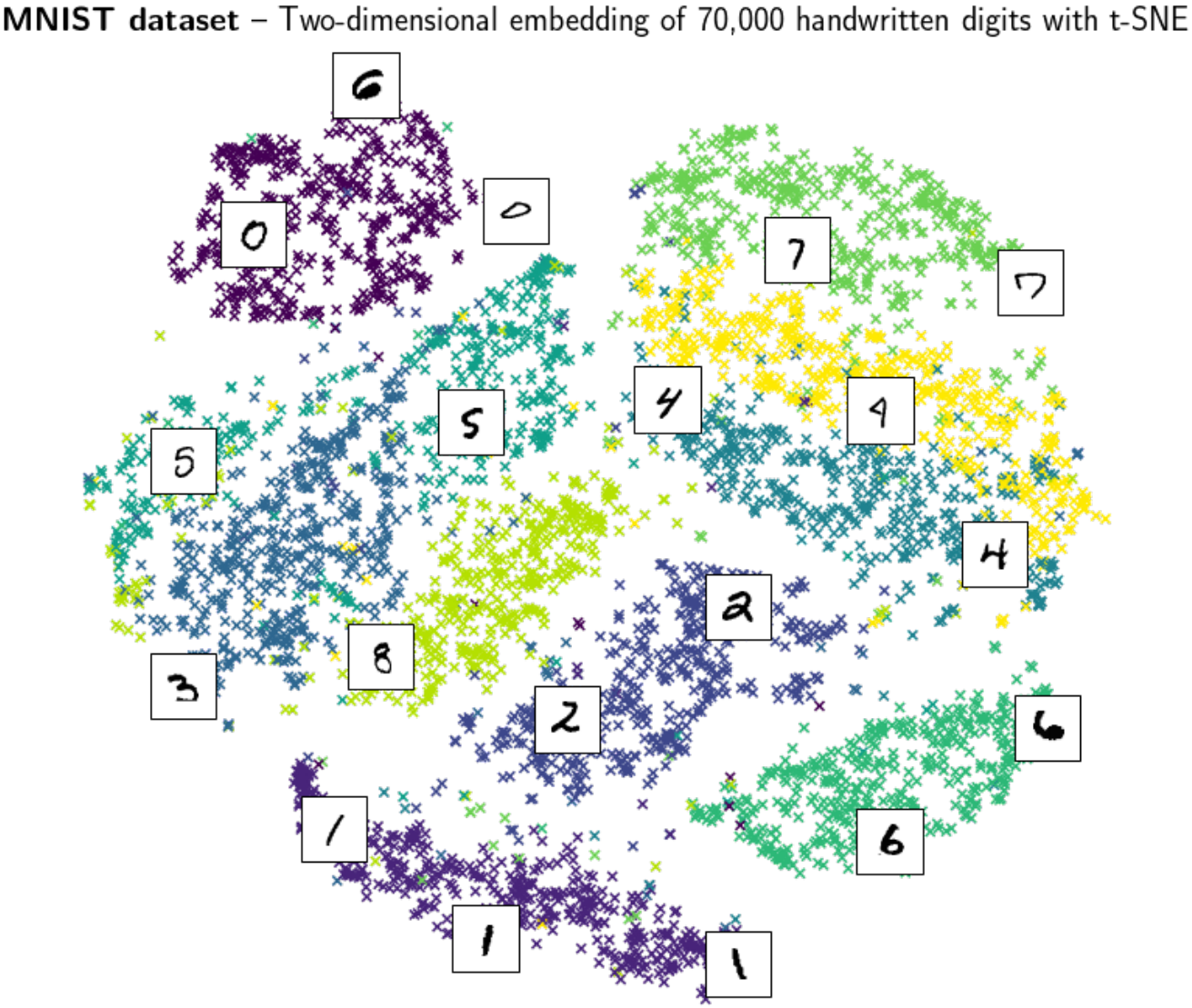
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Algorithm

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations T, learning rate η, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
  compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
  set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
  sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
  for t=*1* **to** T **do**
    compute low-dimensional affinities q_{ij} (using Equation 4)
    compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
    set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
  end
end

Algorithm on MNIST



Algorithm as we implemented it

Data: high dimensional data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\} \subseteq \mathbb{R}^D$, Target Perplexity P, Number of iterations T
Result: low dimensional data $\mathbf{Y} = \{\mathbf{y}_1, \dots, \mathbf{y}_N\} \subseteq \mathbb{R}^d$ such that $d \ll D$ (usually d = 2 or 3)

Part 1

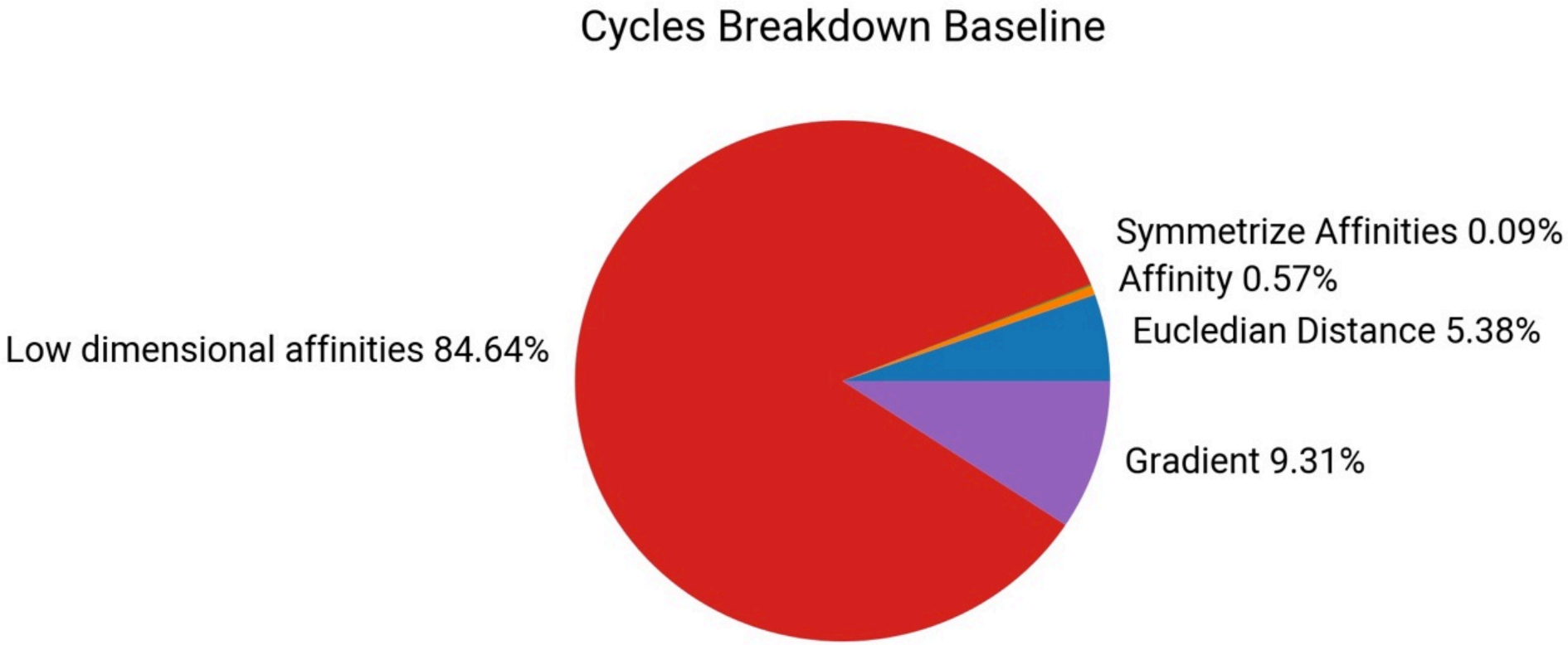
subtract mean $\forall i = 0..N \quad \mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}}$;
recale data $\forall i = 0..N,\ k = 0..D \quad (\mathbf{x}_i)_k = (\mathbf{x}_i)_k \,/\, \max_{i',k'} (\mathbf{x}_{i'})_{k'}$;
compute squared eucledian distances $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$;

Part 2

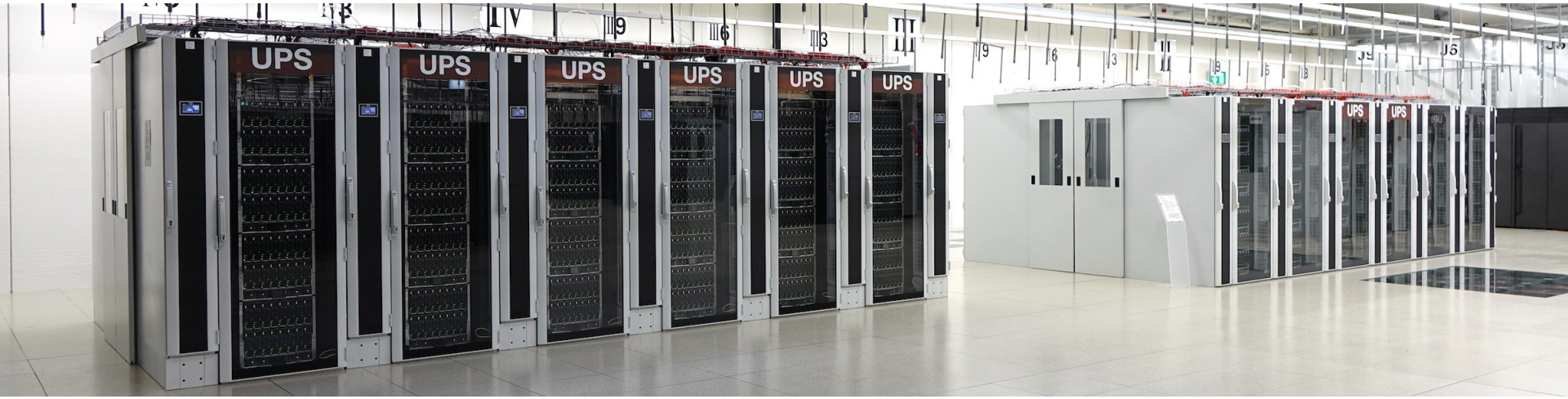
for $i = 0..N$ **do**
  initialize σ_i;
  repeat
    $\forall j = 0..N \quad p_{j|i} = \frac{\exp(-d_{ij}/2\sigma_i^2)}{\sum_{k \neq i} \exp(-d_{ik}/2\sigma_i^2)}$;
    $\text{per}_i = 2^{\sum_j p_{j|i} \log_2 p_{j|i}}$;
  until $per_i = P$;
end
symmetrize $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}$;

Part 3

sample inital $\mathbf{Y}^{(0)} = \{\mathbf{y}_1^{(0)}, \dots, \mathbf{y}_N^{(0)}\}$ from gaussian distribution;
for $t = 1..T$ **do**
  compute $\forall i, j = 0..N \quad q_{ij} = \frac{(1 + \|\mathbf{y}_i^{(t-1)} - \mathbf{y}_j^{(t-1)}\|^2)^{-1}}{\sum_{k \neq l} (1 + \|\mathbf{y}_k^{(t-1)} - \mathbf{y}_l^{(t-1)}\|^2)^{-1}}$;
  compute $\forall i = 0..N \quad \frac{\partial C}{\partial \mathbf{y}_i^{(t-1)}} = 4 \sum_j (p_{ij} - q_{ij})(\mathbf{y}_i^{(t-1)} - \mathbf{y}_j^{(t-1)})(1 + \|\mathbf{y}_i^{(t-1)} - \mathbf{y}_j^{(t-1)}\|^2)^{-1}$;
  update $\forall i = 0..N \quad \mathbf{y}_i^{(t)} = \mathbf{y}_i^{(t-1)} + \eta \frac{\partial C}{\partial \mathbf{y}_i^{(t-1)}} + \alpha(t)(\mathbf{y}_i^{(t-1)} - \mathbf{y}_i^{(t-2)})$;
  subtract mean $\forall i = 0..N \quad \mathbf{y}_i^{(t-1)} = \mathbf{y}_i^{(t-1)} - \bar{\mathbf{y}}^{(t-1)}$;
end


Cycles Breakdown Baseline
Symmetrize Affinities 0.09%
Affinity 0.57%
Eucledian Distance 5.38%
Gradient 9.31%
Low dimensional affinities 84.64%

Experimental Setup



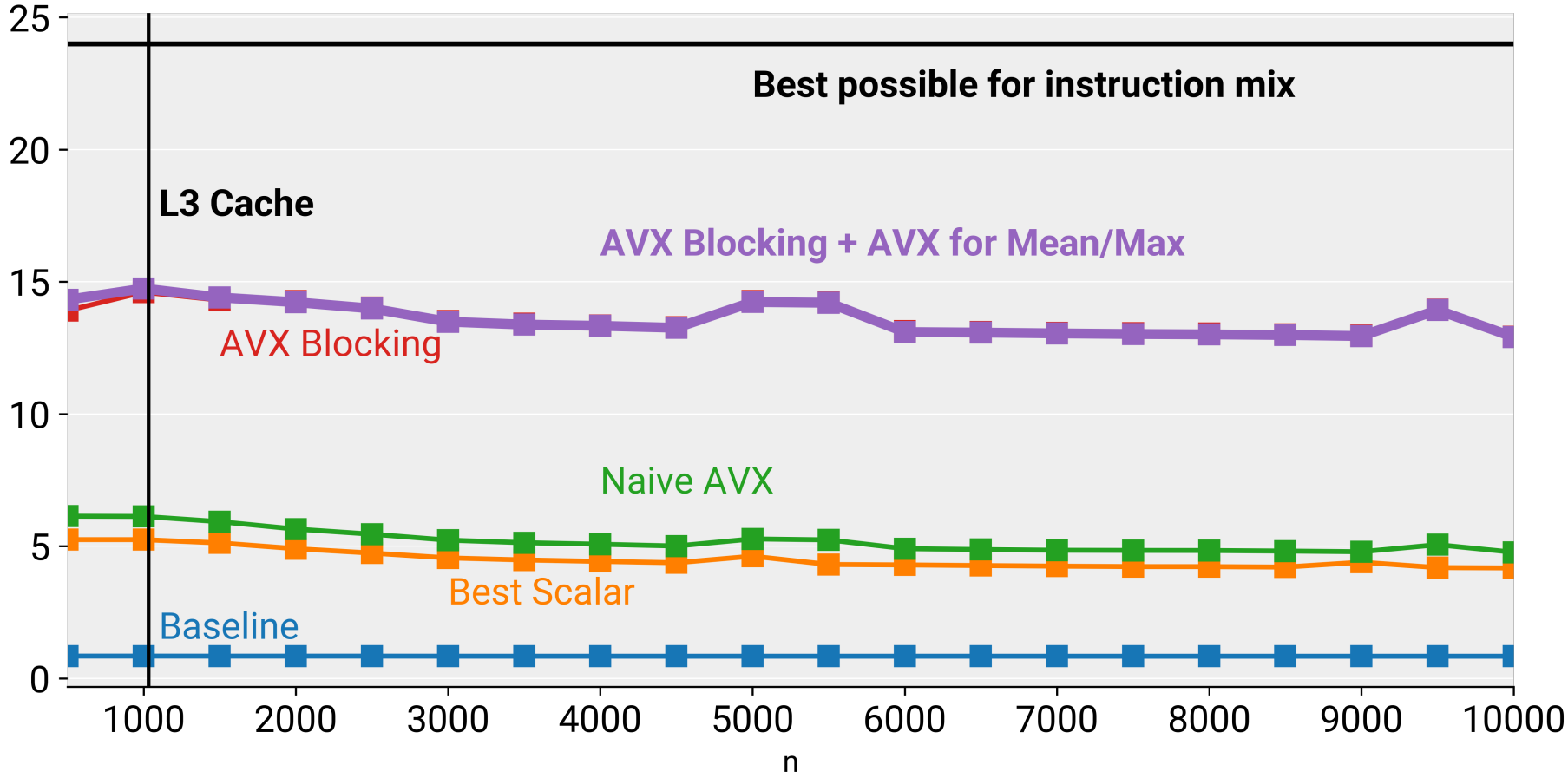
- **Euler III:** Intel Xeon E5-1585Lv5 3.0 GHz
- **Architecture:** Skylake (AVX2.0)
- **Cache:**
  - L1 Cache: 32 KB
  - L2 Cache: 256 KB
  - L3 Cache: 8 MB
- **Compiler:** ICC 16.0

Part 1: Data preprocessing

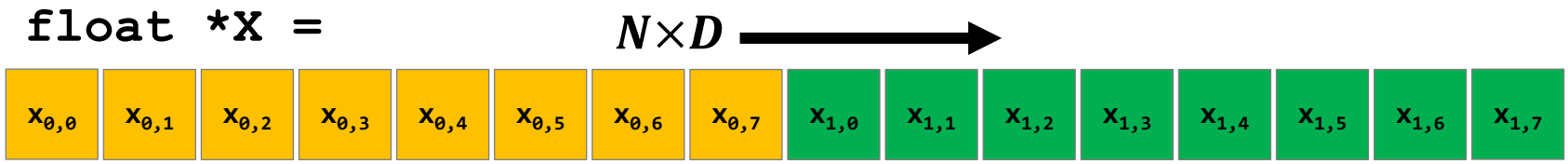
$$I \approx \frac{3\,DN + D + 3DN(N-1)/2\ \ flops}{4(N^2 + ND)\ bytes}$$

subtract mean $\forall i = 0..N \quad \mathbf{x}_i = \mathbf{x}_i - \bar{\mathbf{x}}$;
recale data $\forall i = 0..N,\ k = 0..D \quad (\mathbf{x}_i)_k = (\mathbf{x}_i)_k \,/\, \max_{i',k'} (\mathbf{x}_{i'})_{k'}$;
compute squared eucledian distances $d_{ij} = \|\mathbf{x}_i - \mathbf{x}_j\|^2$;

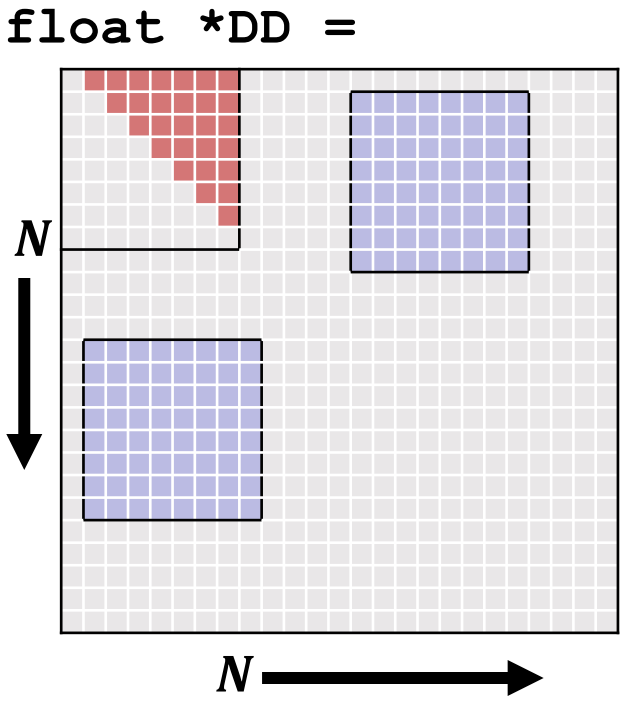
Performance, D = 1000



Part 1: Optimized Version, Distance




```
for i=0:8:N
  for j=i+8:8:N
    for i'=i:i+8
      accum0[0:7] = 0 ... accum7[0:7] = 0
      for k=0:8:D
        dist0[0:7] = X[i][k:k+7]
                   - X[j][k:k+7]
        accum0[0:7] += dist0*dist0
        ...
        dist7[0:7] = X[i+7][k:k+7]
                   - X[j+7][k:k+7]
        accum7[0:7] += dist7*dist7
      Reduce 8 accums into res[0:8]
      Multiply by factor
      Store res as 1 row in upper block
    Transpose 8 rows and store lower block
```



Part 2: Symmetrize affinities

$$p_{ij} = \frac{p_{ij} + p_{ji}}{2}$$

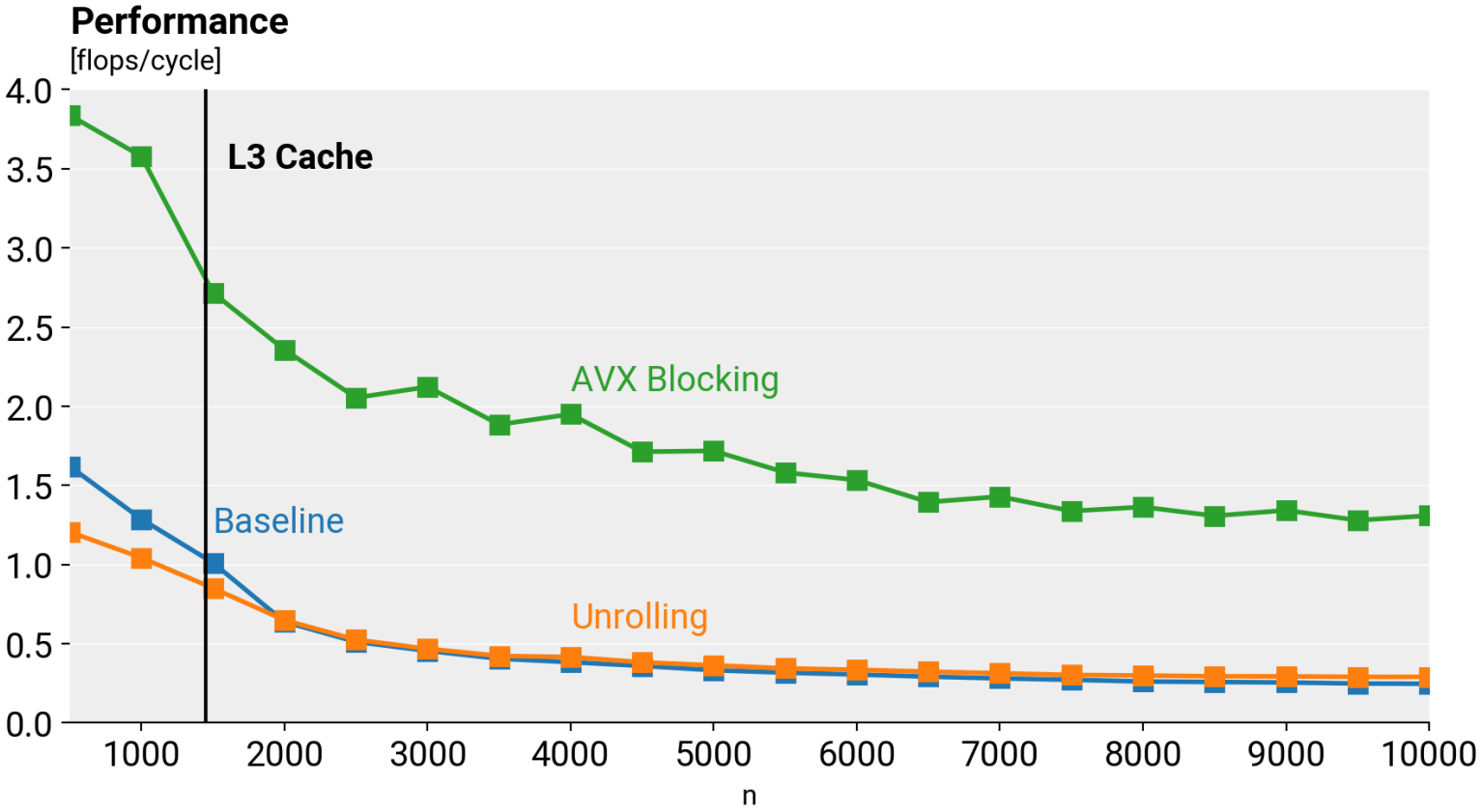
+ normalize per row

For the Affinity calculation: no improvement

Symmetrize affinities:

Computation is compute bound for small N: $I(N)\ \epsilon\ O(N^2)$) for N <= 1000

Computation is memory bound for larger N: $I(N)\ \epsilon\ O(\frac{1}{N})$



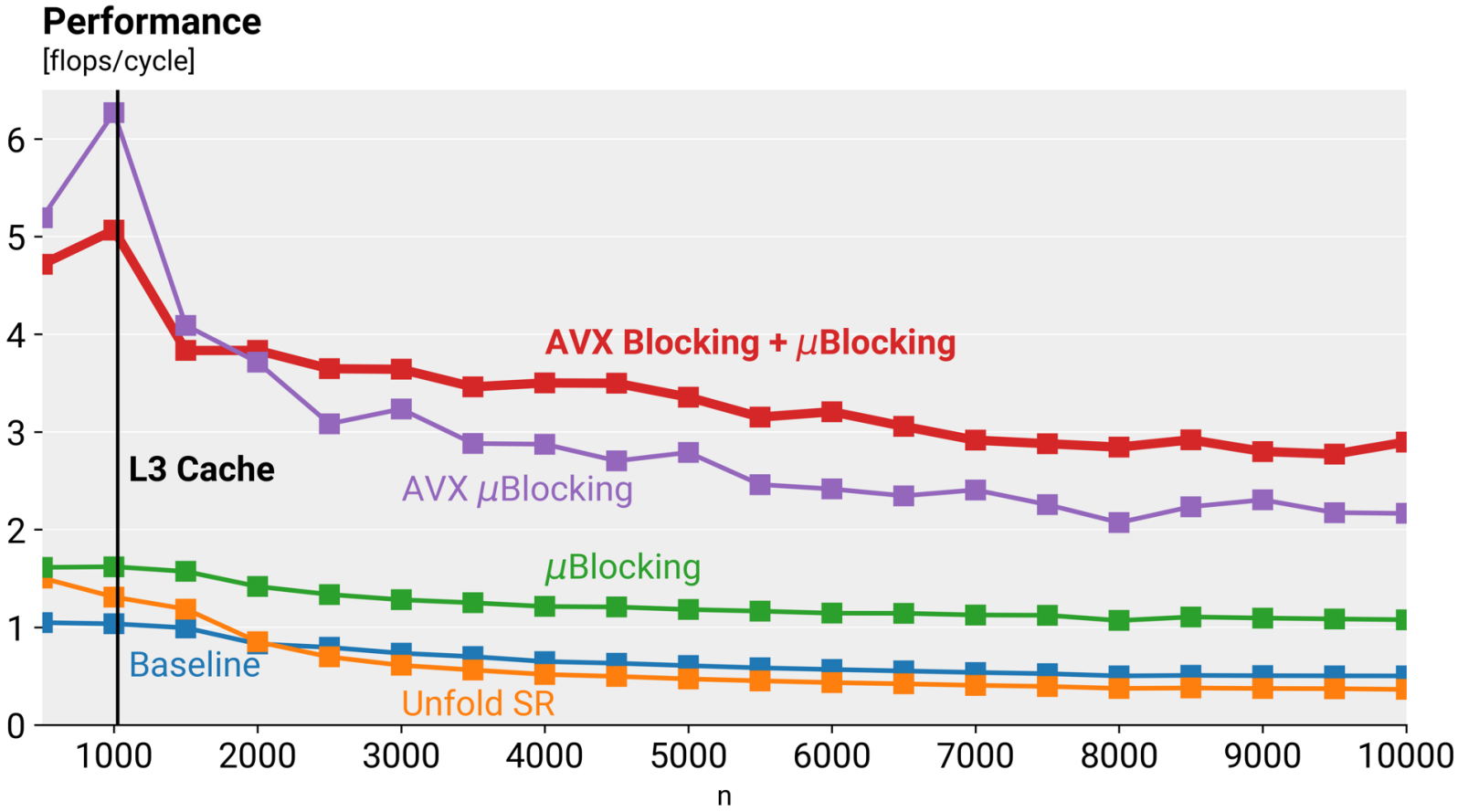
Part 3: Training Loop

Computation Low Dimensional Affinities

$$t_{ij} = (1 + ||y_i - y_j||^2)^{-1}$$

$$I(n) = \frac{8\,N(N-1)/2\ \ flops}{4(N^2 + 2N)bytes} \approx 1\ flop/byte$$

$$T = \sum_{k \neq l} t_{kl}$$



Part 3: Training Loop

Gradient Computation, Update and Normalize

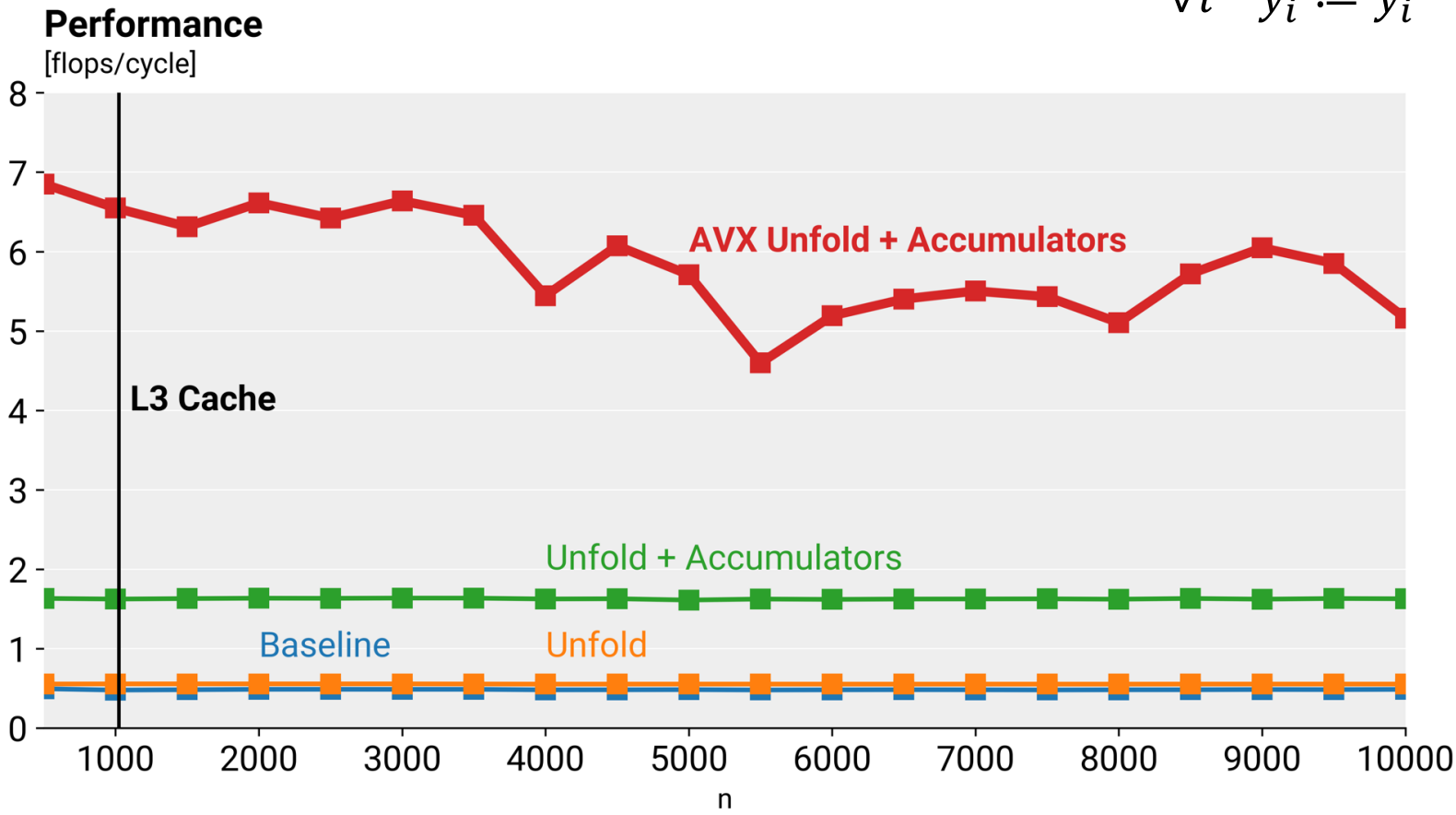
$$\frac{\partial C}{\partial y_i} = \sum_j \left(p_{ij} - \frac{t_{ij}}{T}\right)(y_i - y_j)t_{ij}$$

$$\Delta y_i^{(t)} = \eta\, \Delta y_i^{(t-1)} - \alpha \frac{\partial C}{\partial y_i}$$

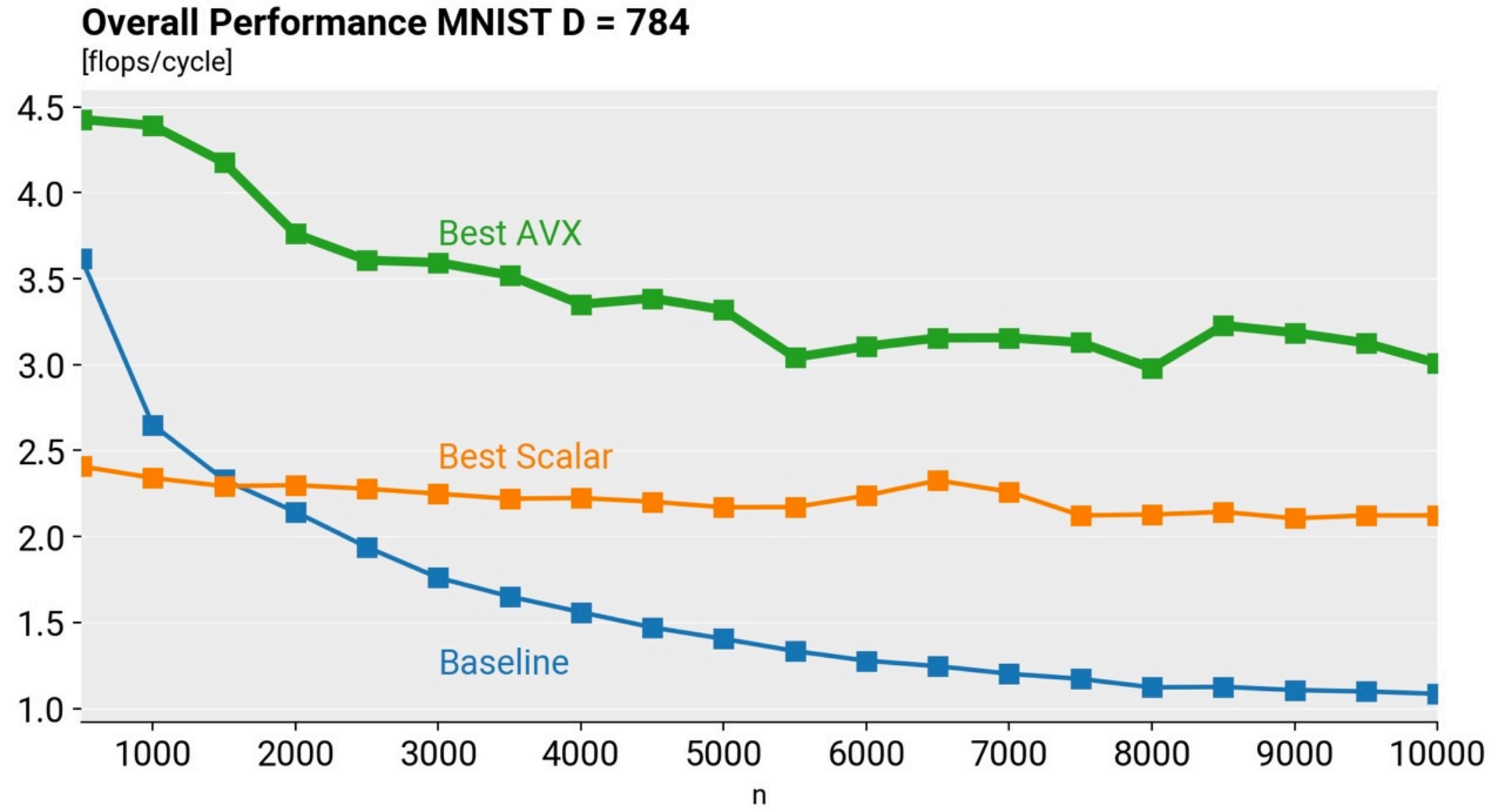
$$y_i^{(t)} = y_i^{(t-1)} + \Delta y_i^{(t)}$$

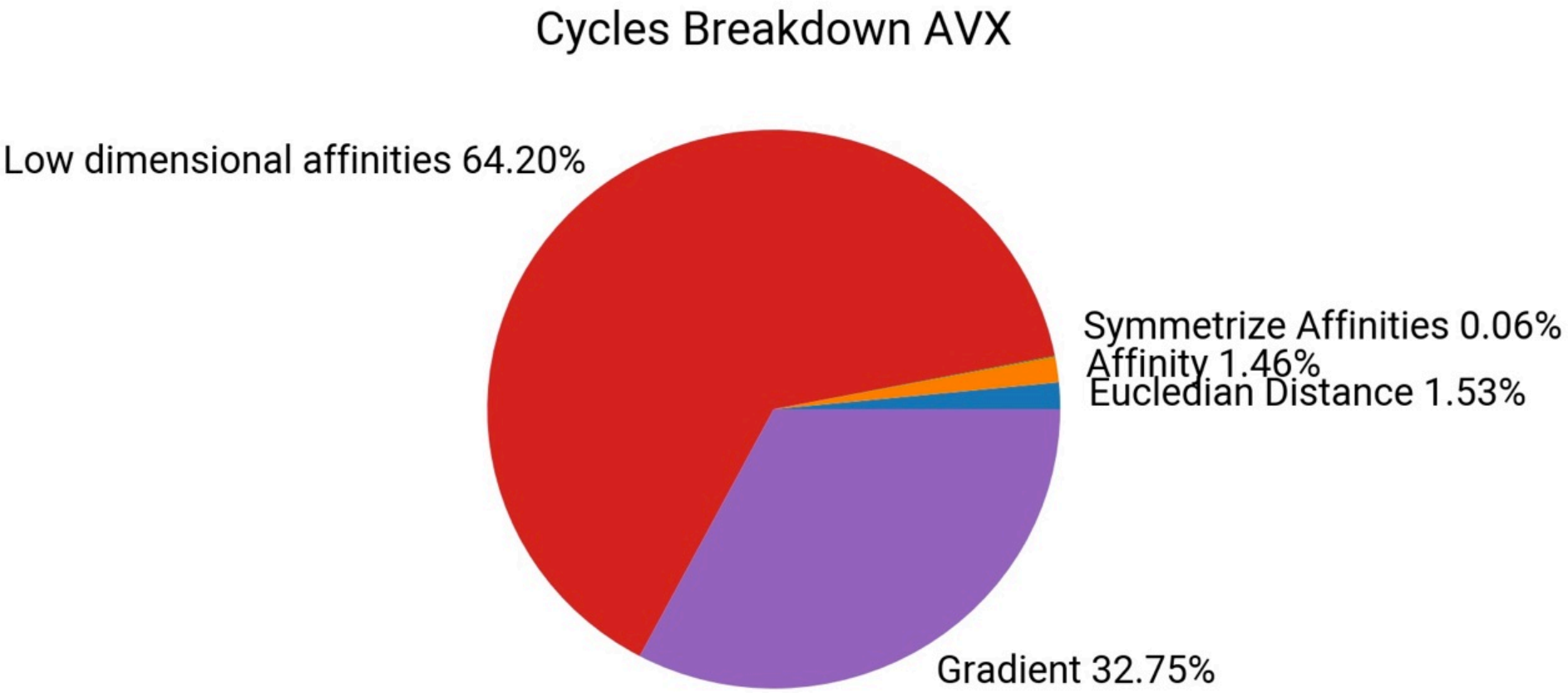
$$\forall i \quad y_i := y_i - \hat{y}$$

$$I(n) = \frac{9N^2 + 4N + 2flops}{4(2N^2 + 4N)bytes} \approx \frac{9}{8}\ flop/byte$$



Overall


Overall Performance MNIST D = 784
[flops/cycle]
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
Best AVX
Best Scalar
Baseline
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
n


Cycles Breakdown AVX
Low dimensional affinities 64.20%
Symmetrize Affinities 0.06%
Affinity 1.46%
Eucledian Distance 1.53%
Gradient 32.75%