Exploring Machine Learning Capabilities in Chapel

Iain Moncrief

Oregon State University

gh://iainmon

Summer 2023 Internship

How close can we get to TensorFlow.chpl or ChaTorch?

if you don't already know Chapel and have only 10 weeks, then not very close

Primitive machine learning library \ {auto differentiation}

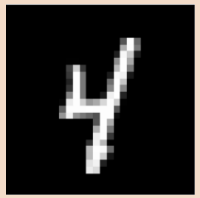
`Dense,Conv,ReLU,SoftMax,...` layer types

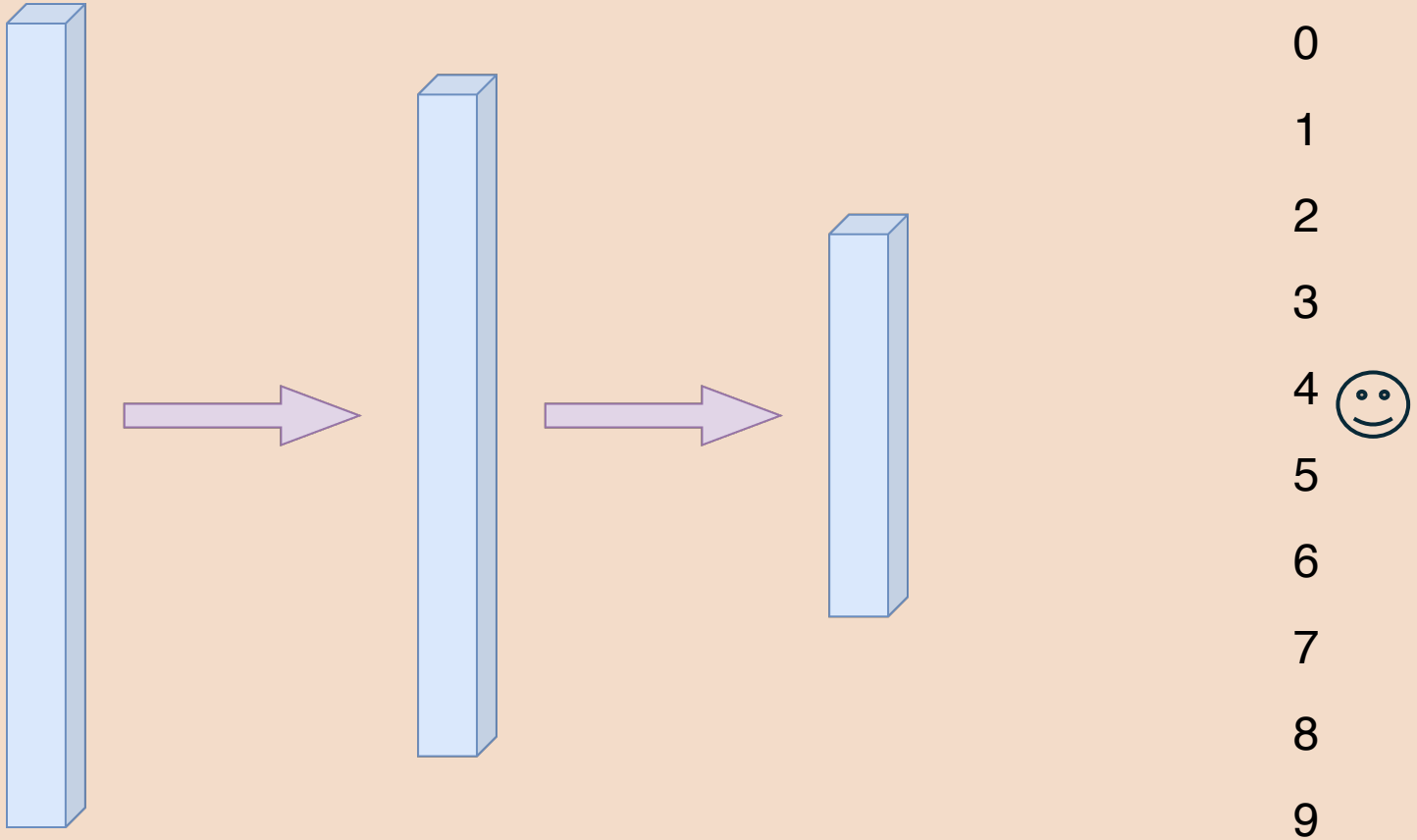
MNIST CNN & perceptron classification

Parallelized batched back propagation

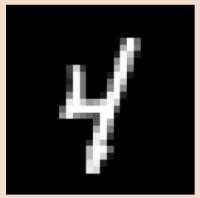
*1 more week and it would have been distributed

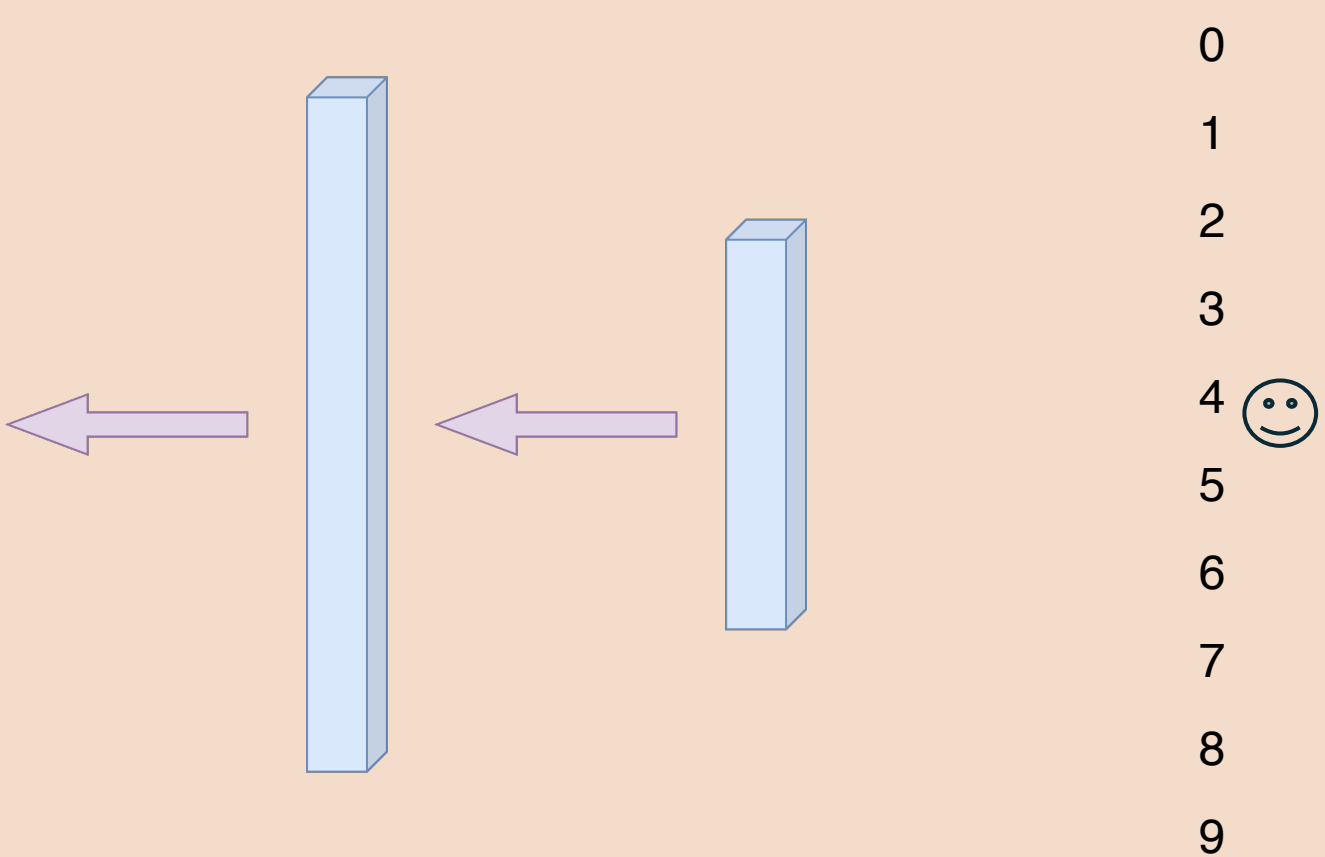
Background



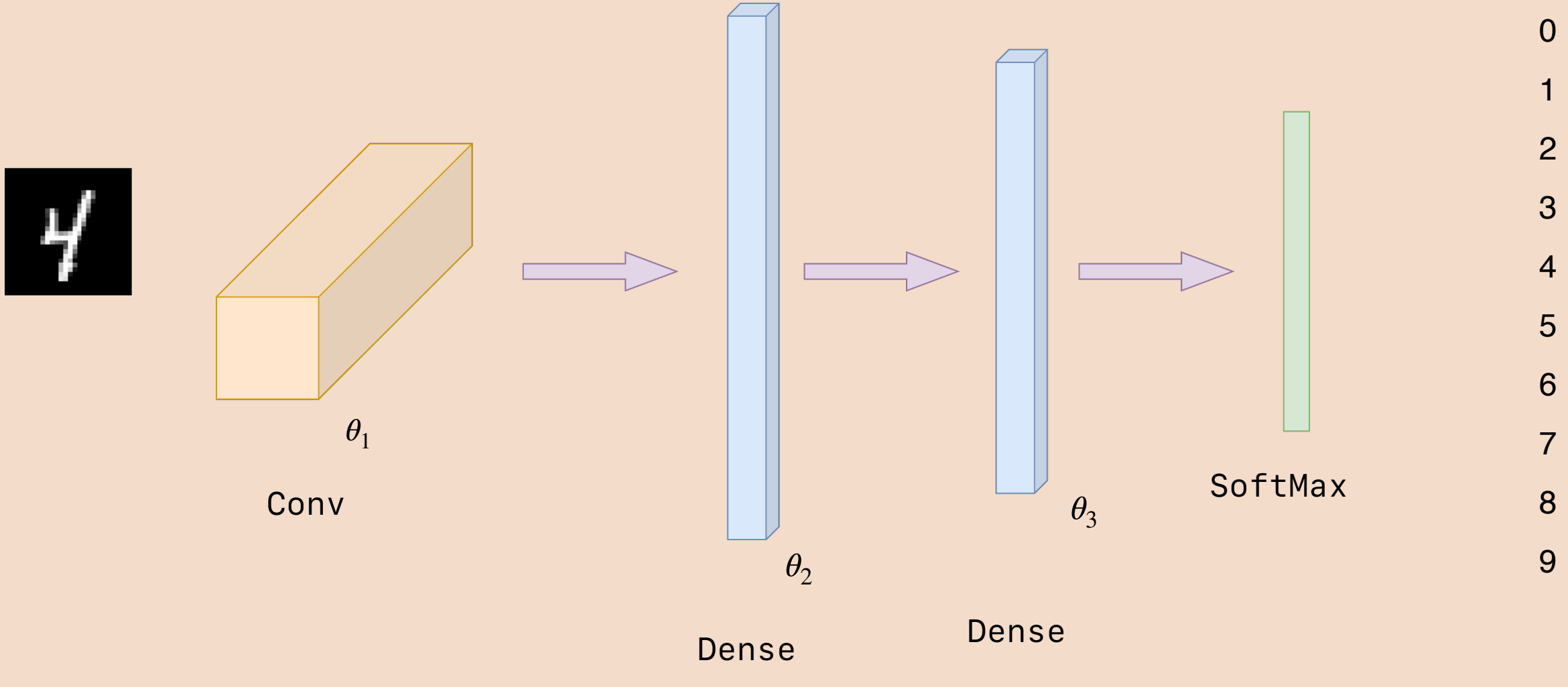


Background

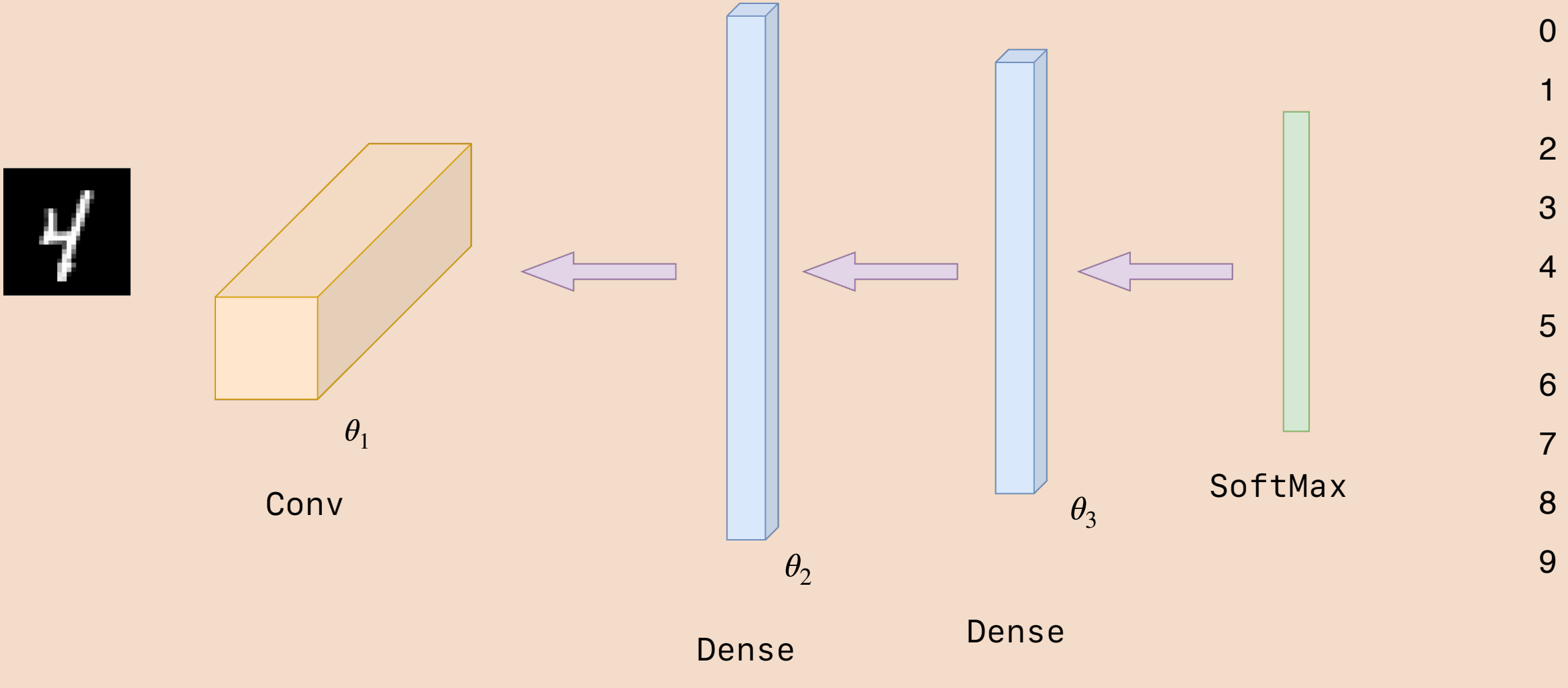




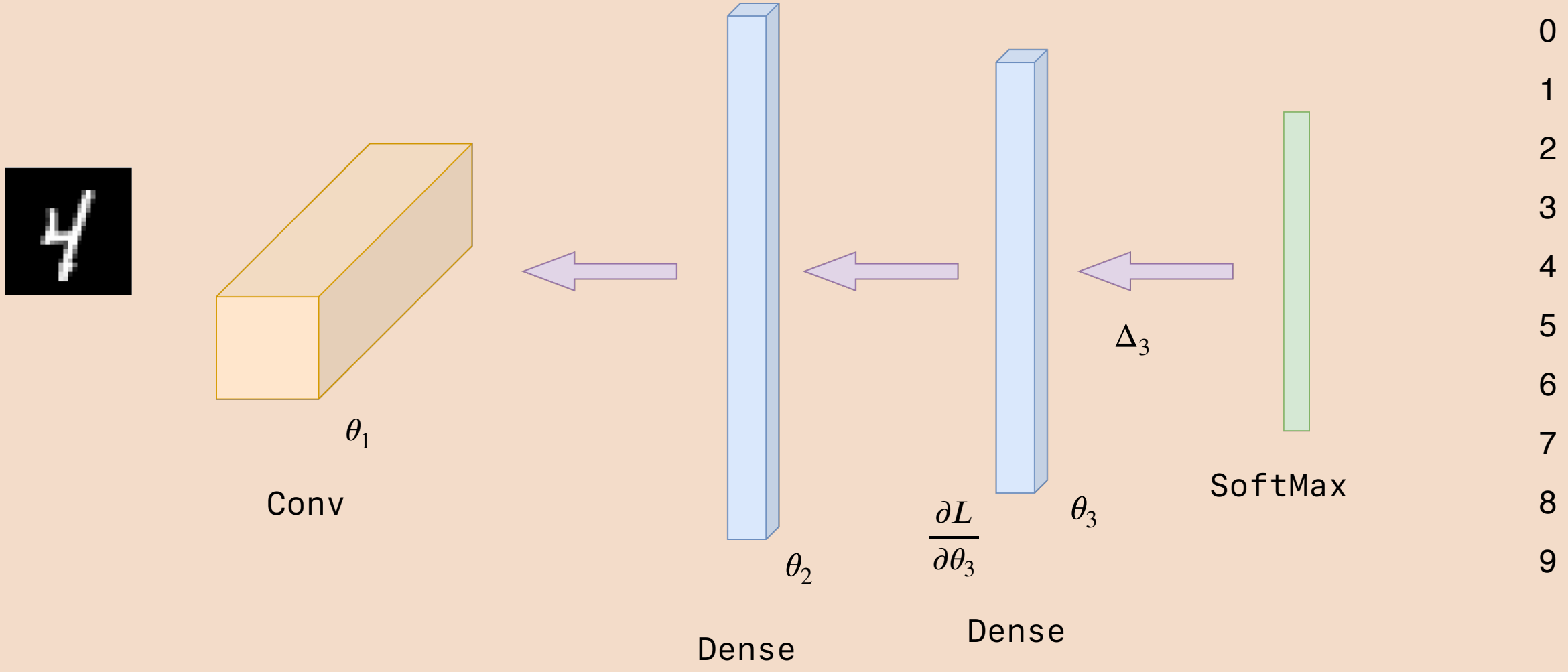
Representation in Chapel



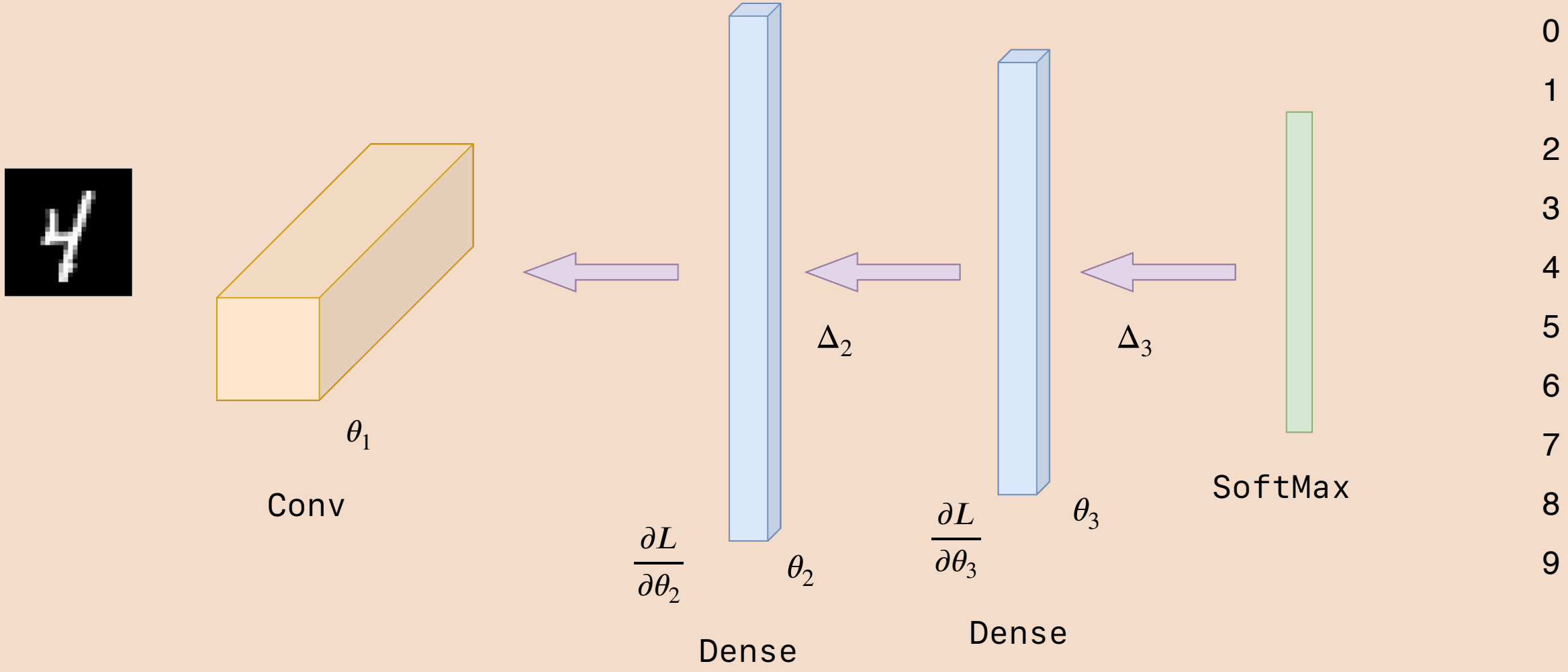
Representation in Chapel



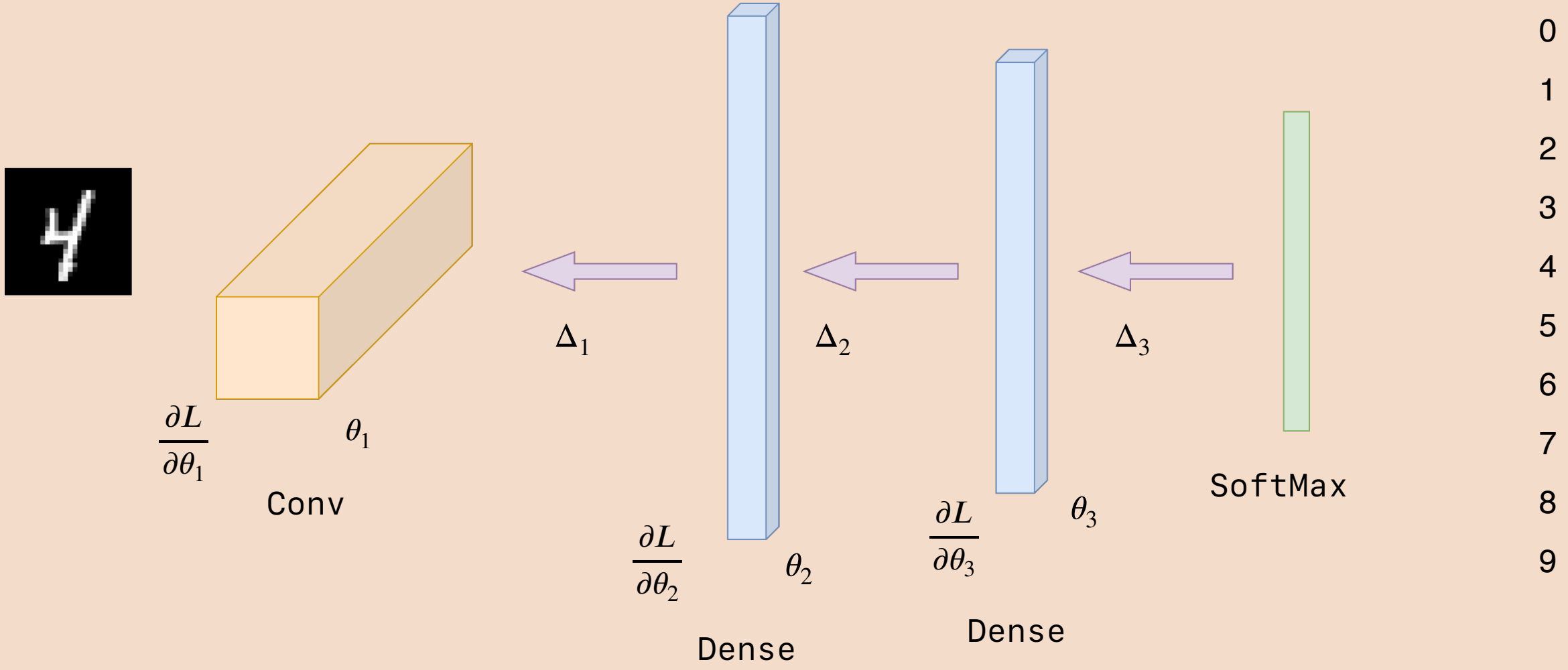
Representation in Chapel



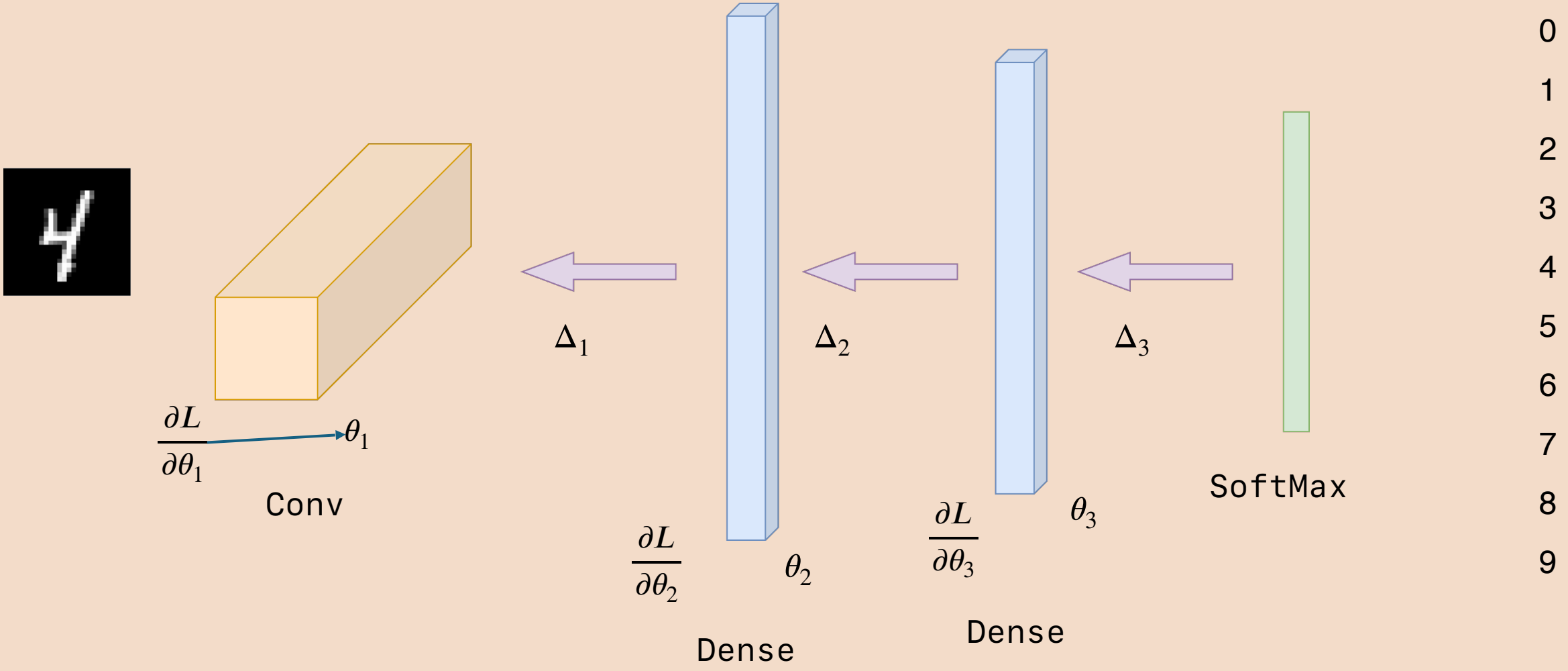
Representation in Chapel



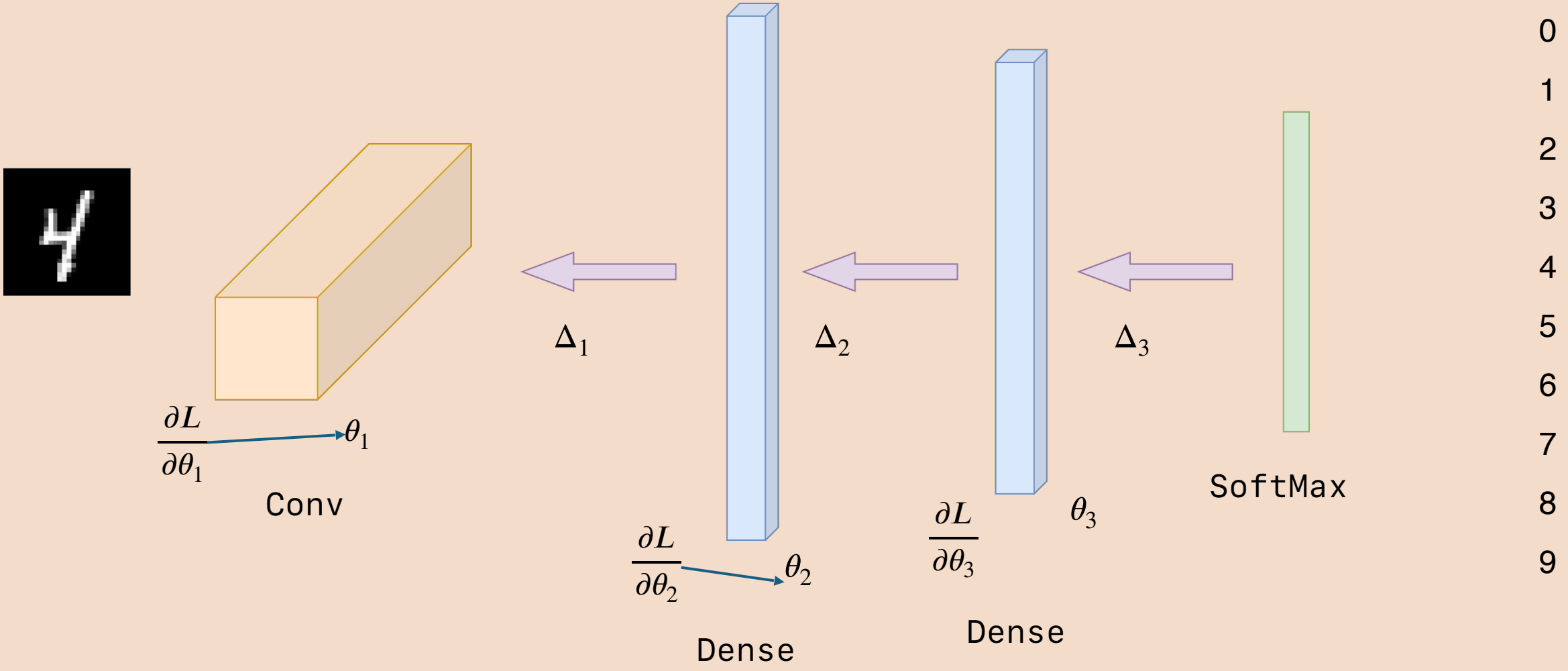
Representation in Chapel



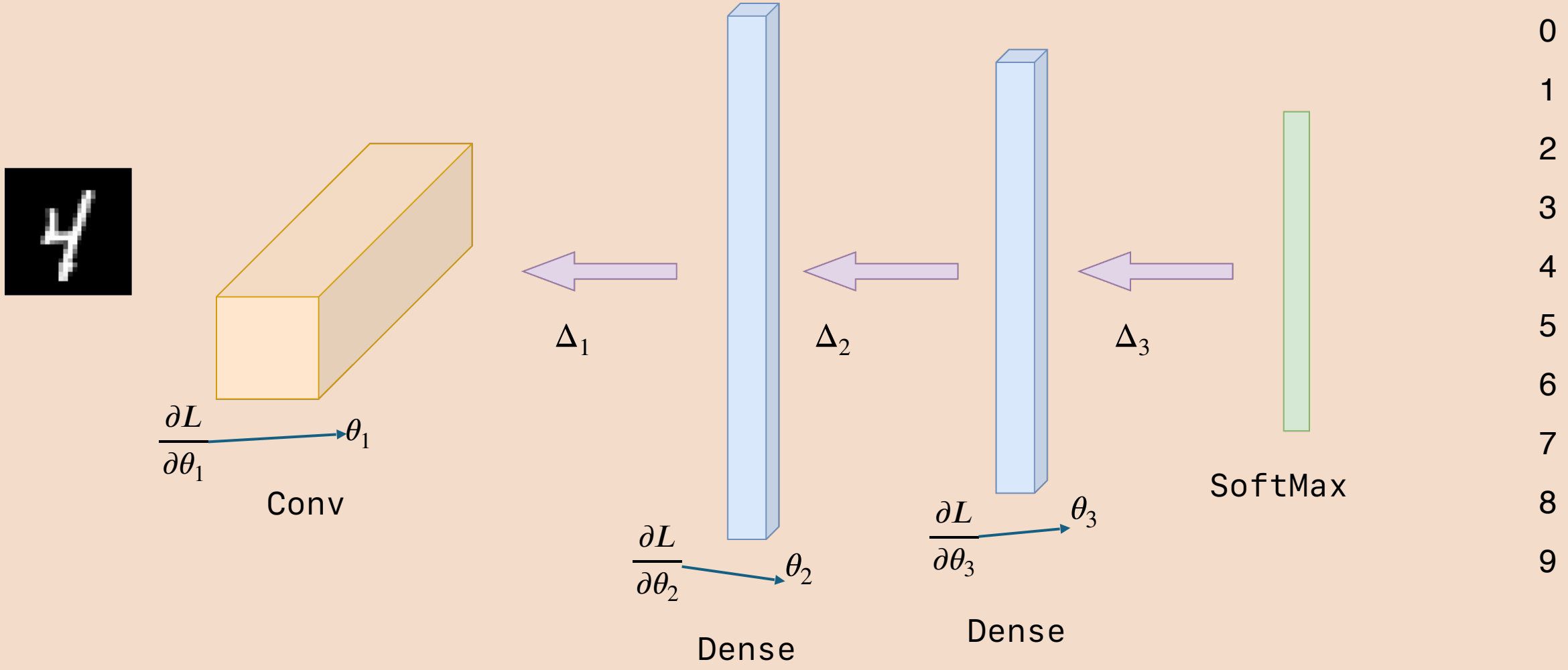
Representation in Chapel

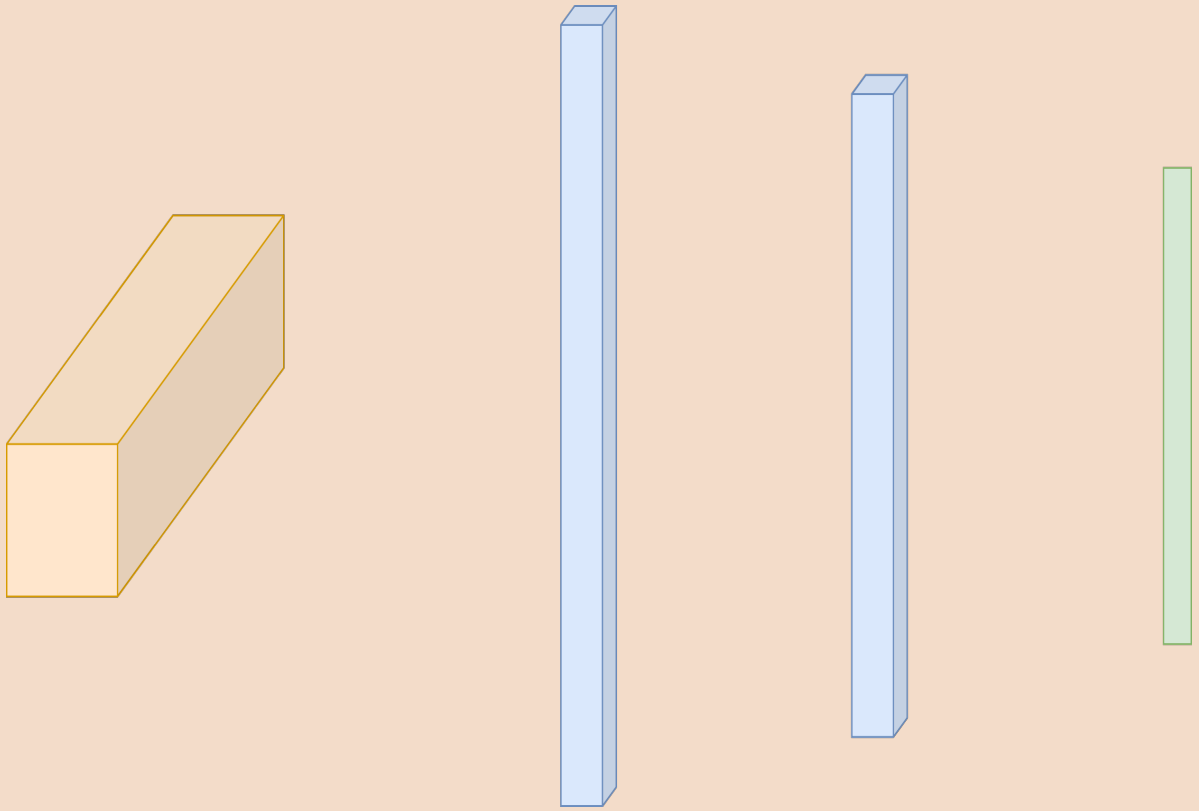


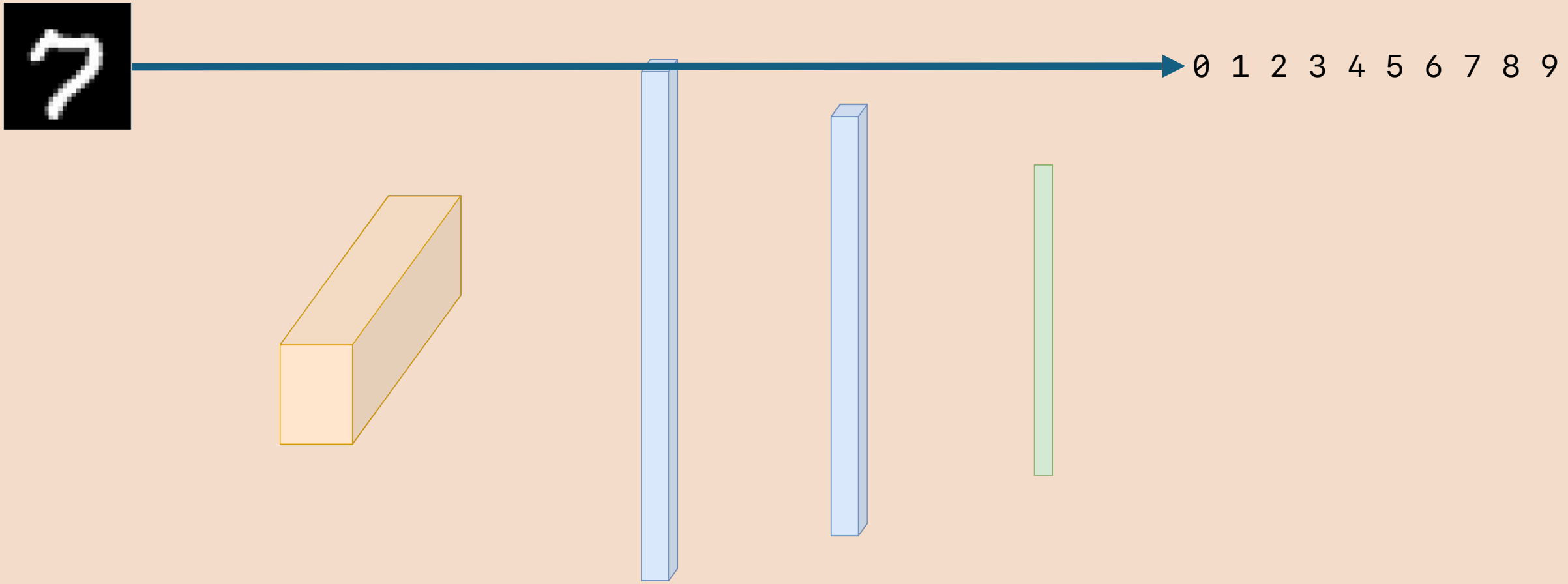
Representation in Chapel

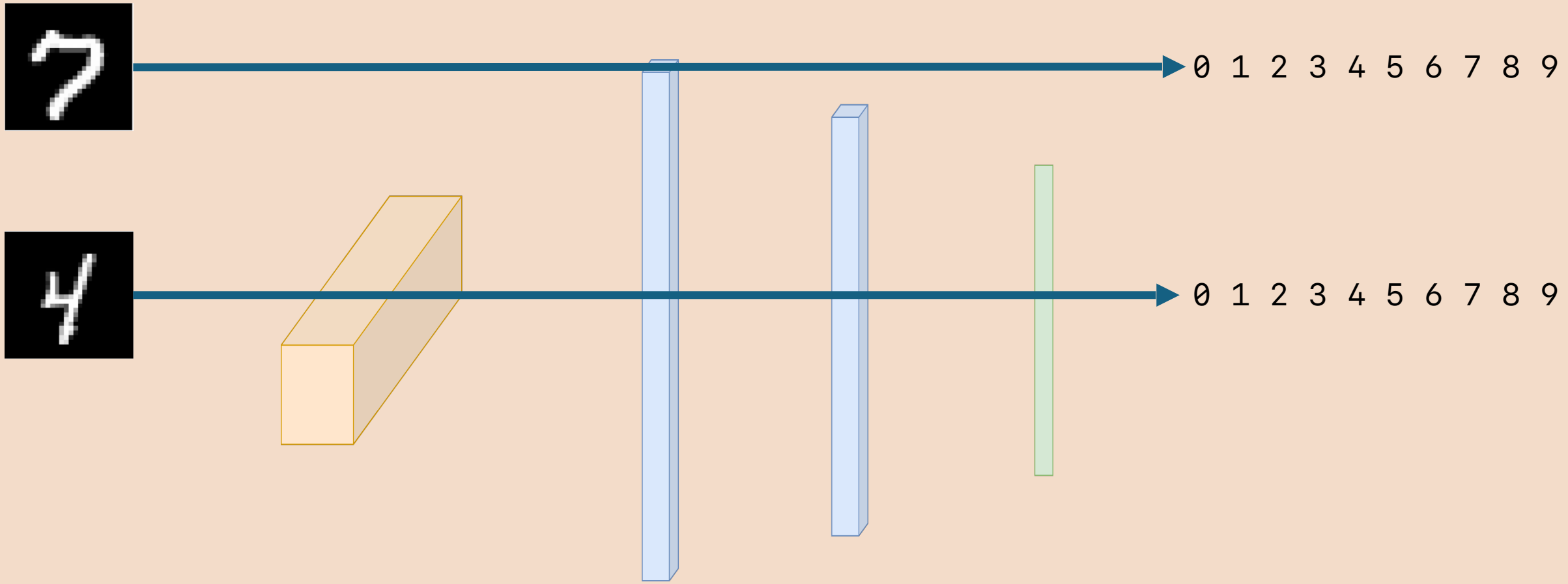


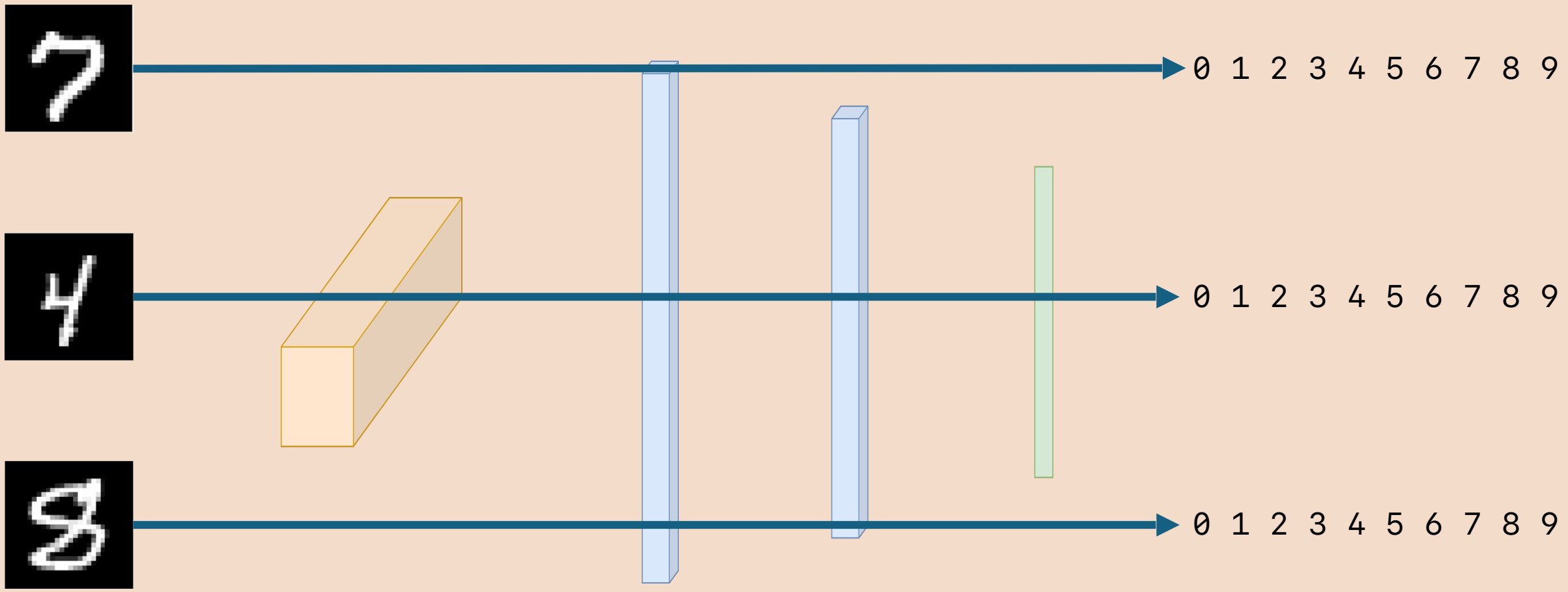
Representation in Chapel





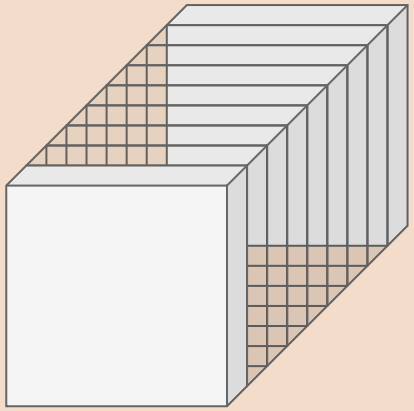

0 1 2 3 4 5 6 7 8 9


0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

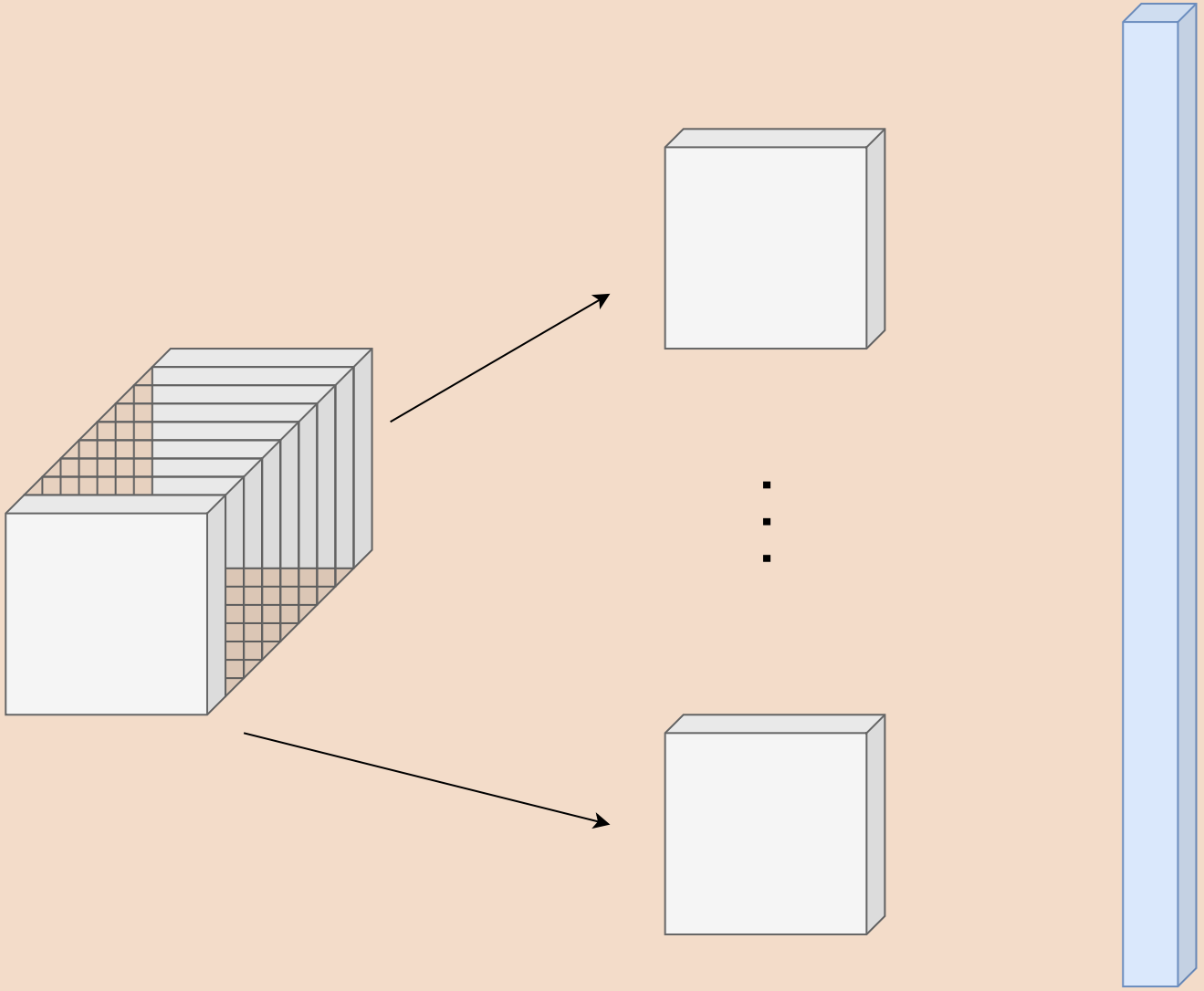

0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9
0 1 2 3 4 5 6 7 8 9

Chapel Parallelism

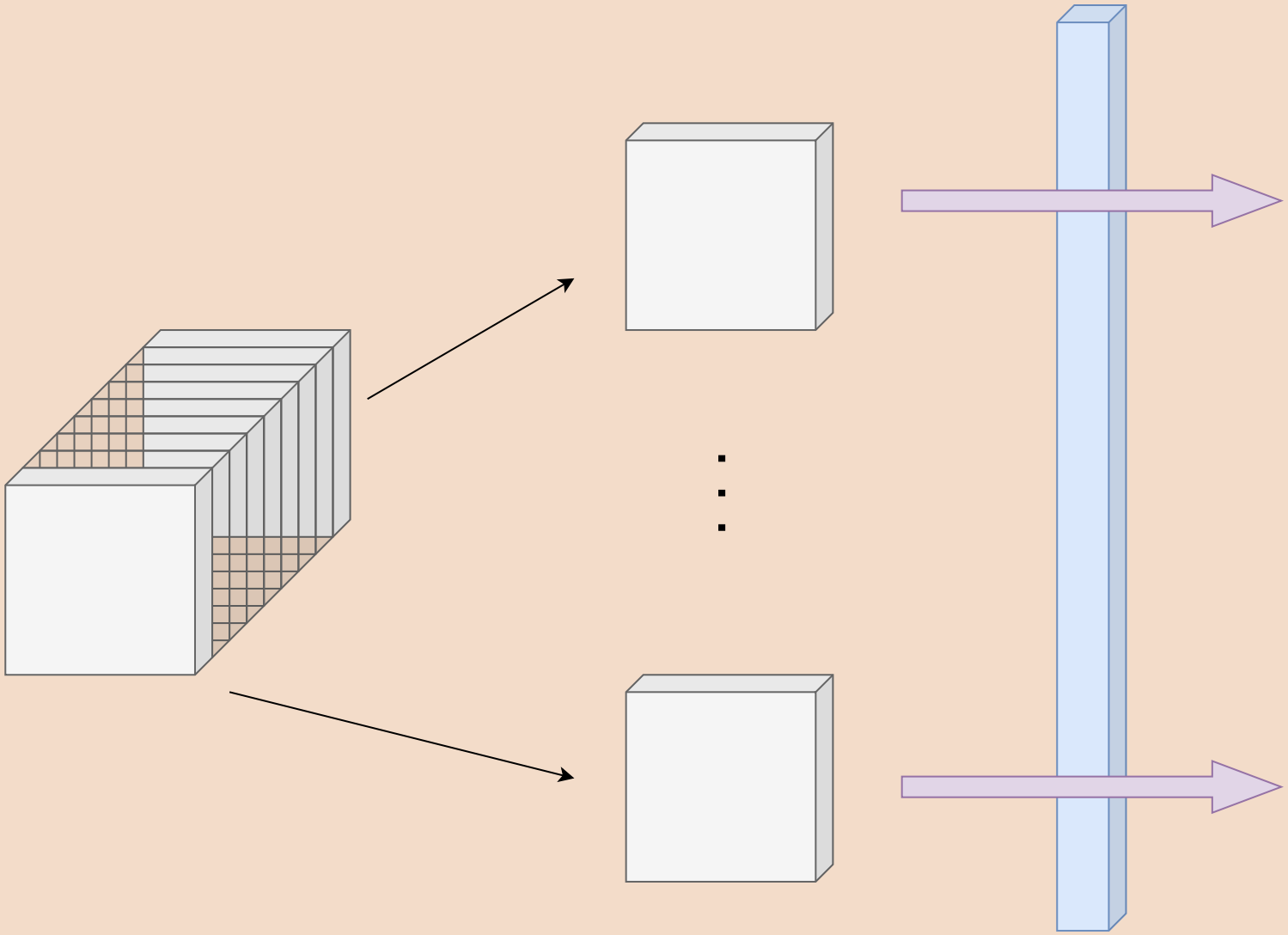
Chapel Parallelism



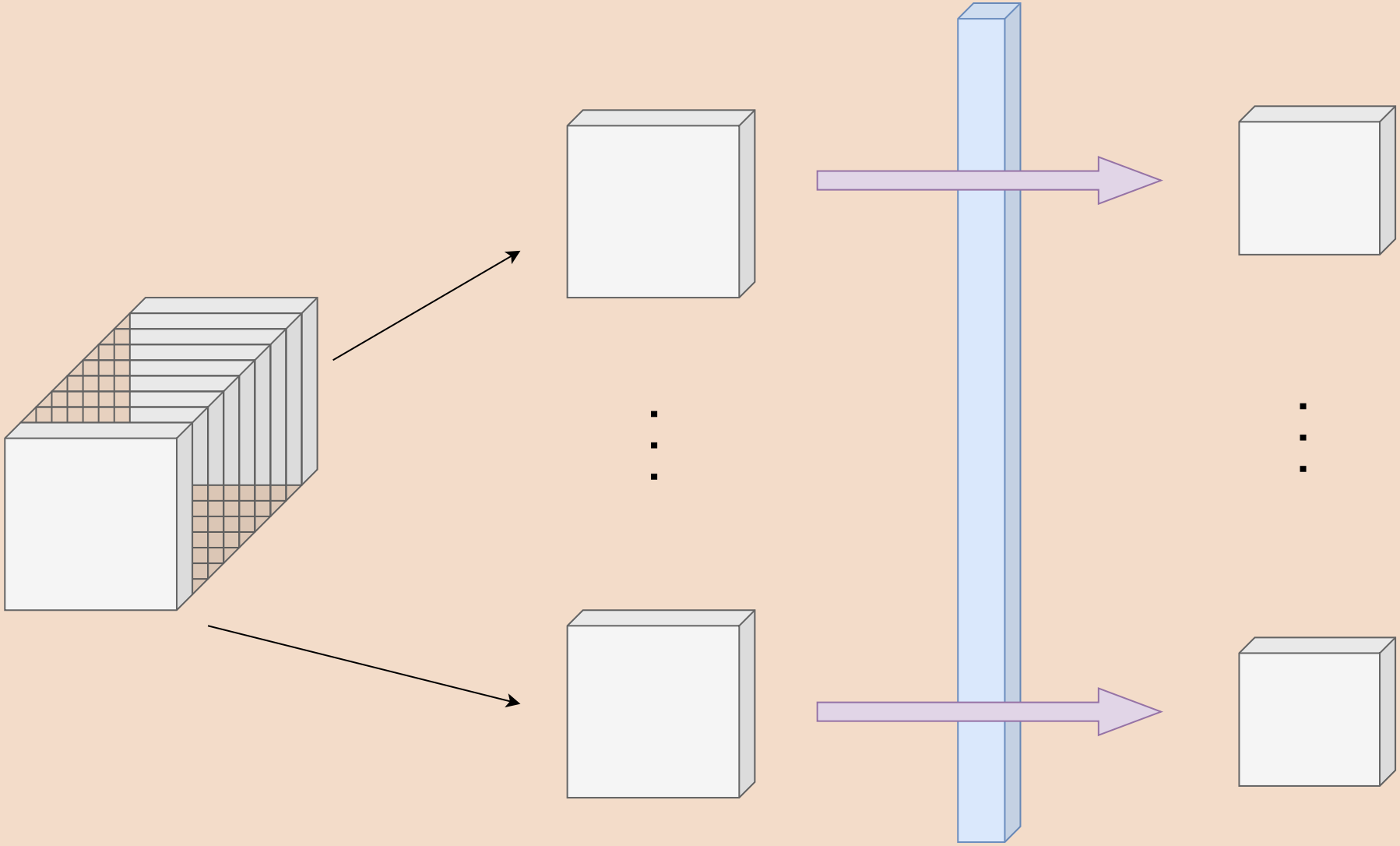
Chapel Parallelism



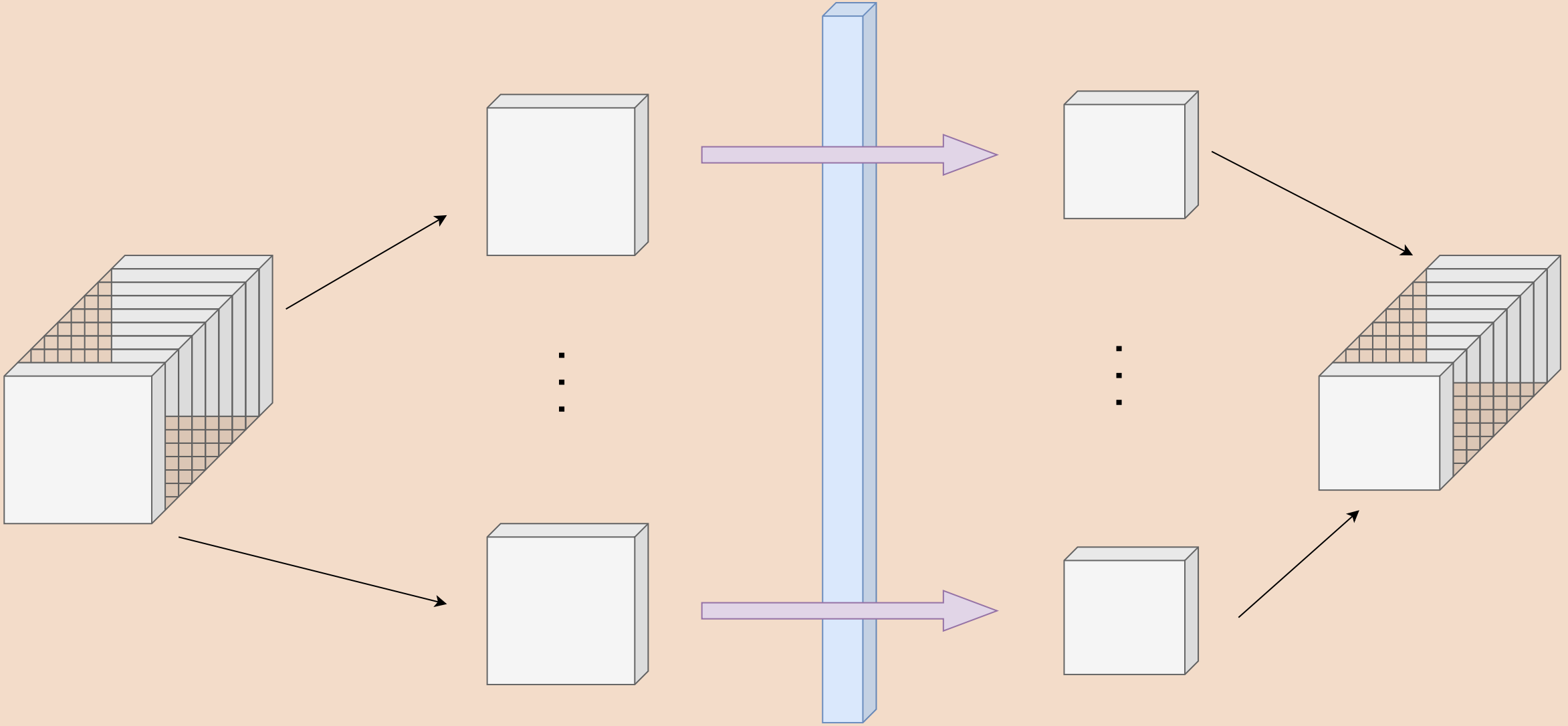
Chapel Parallelism



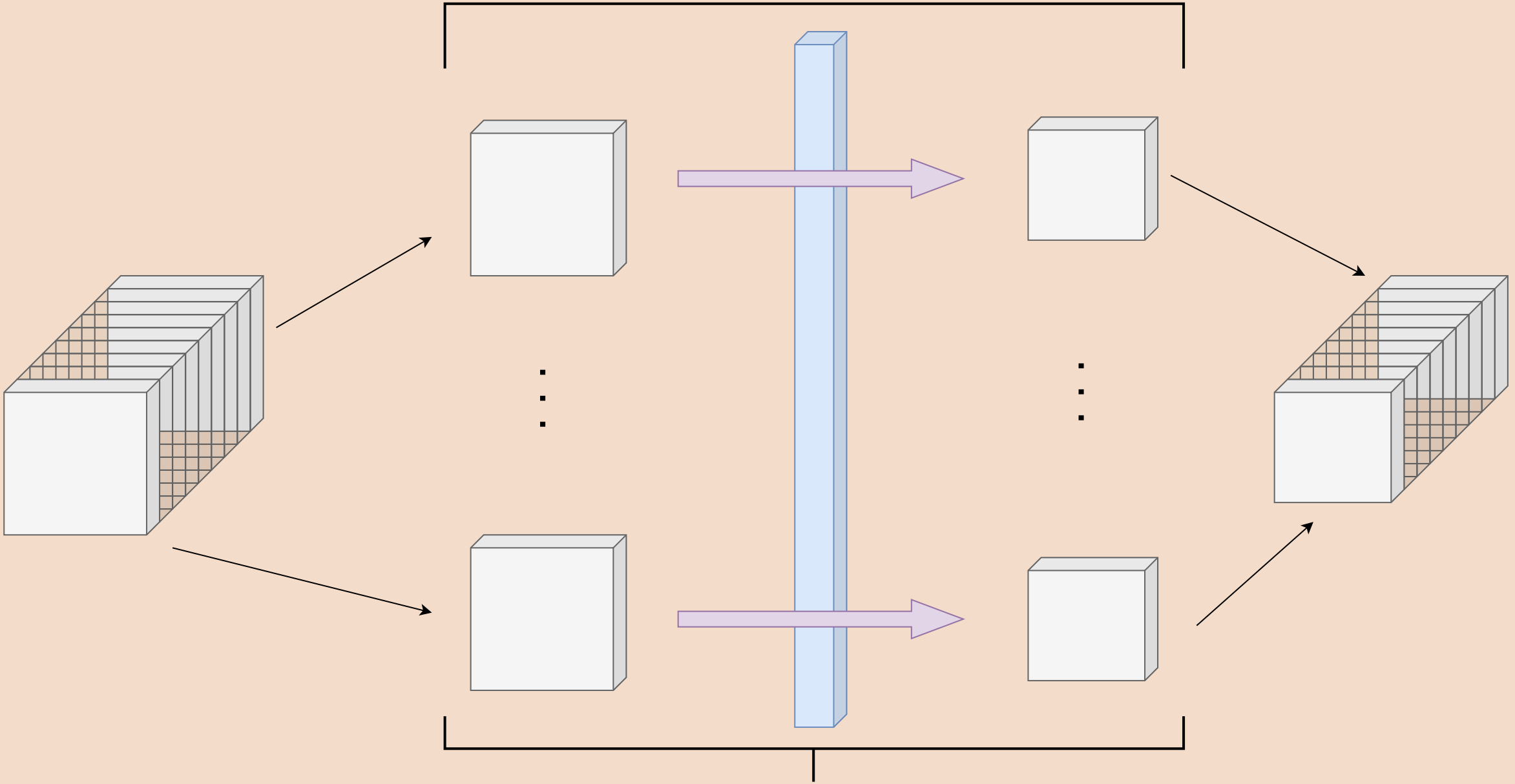
Chapel Parallelism

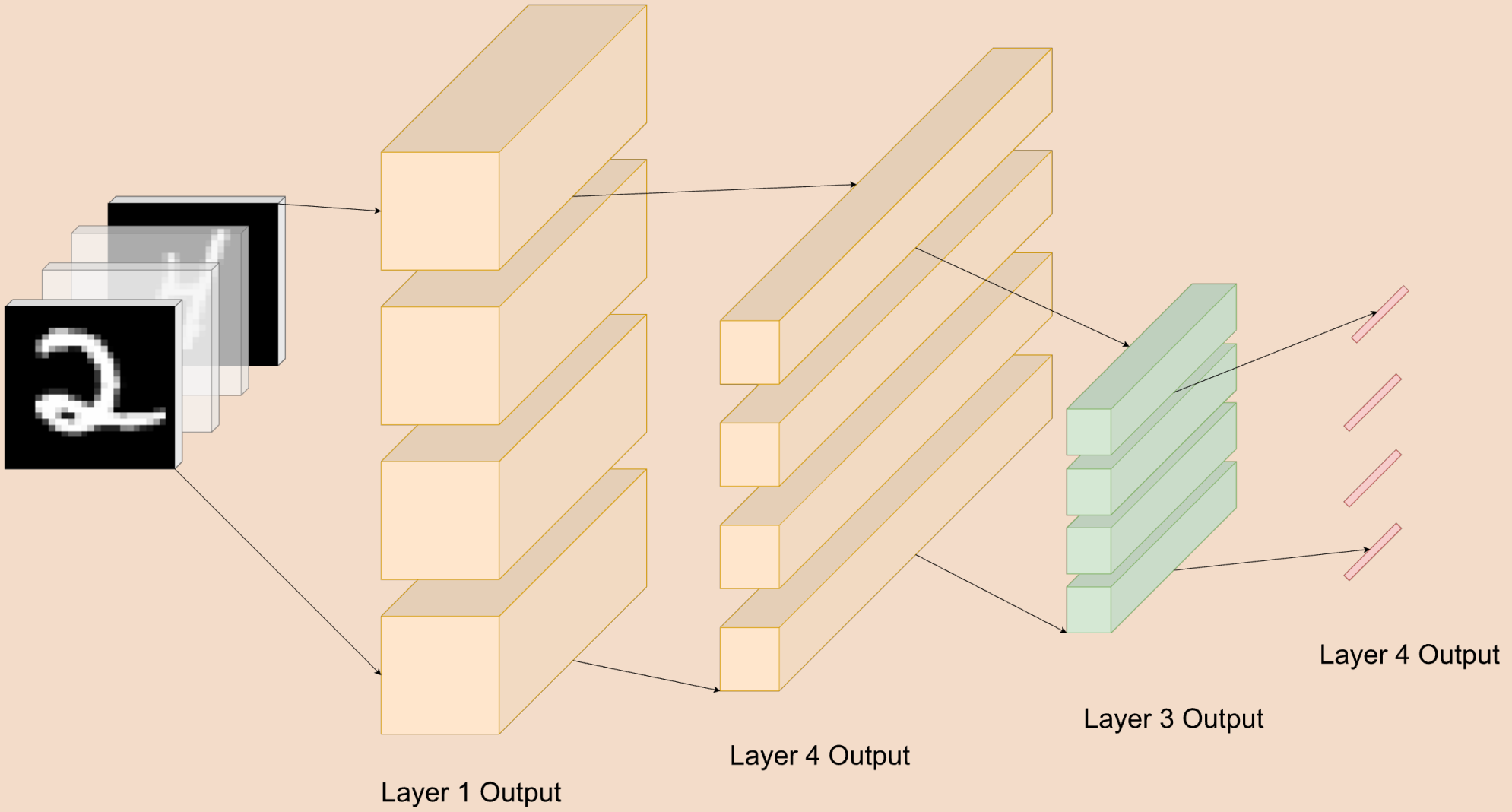


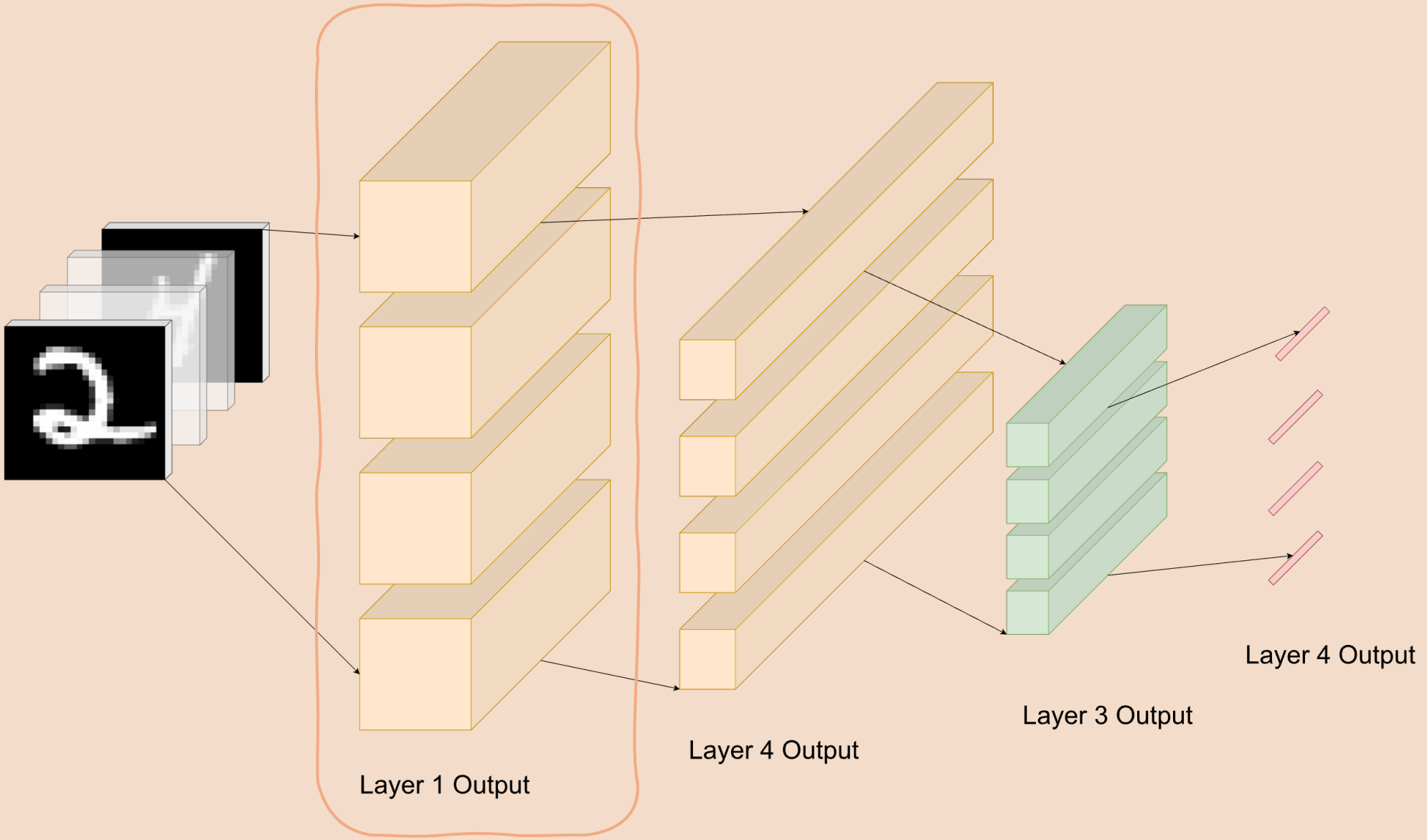
Chapel Parallelism

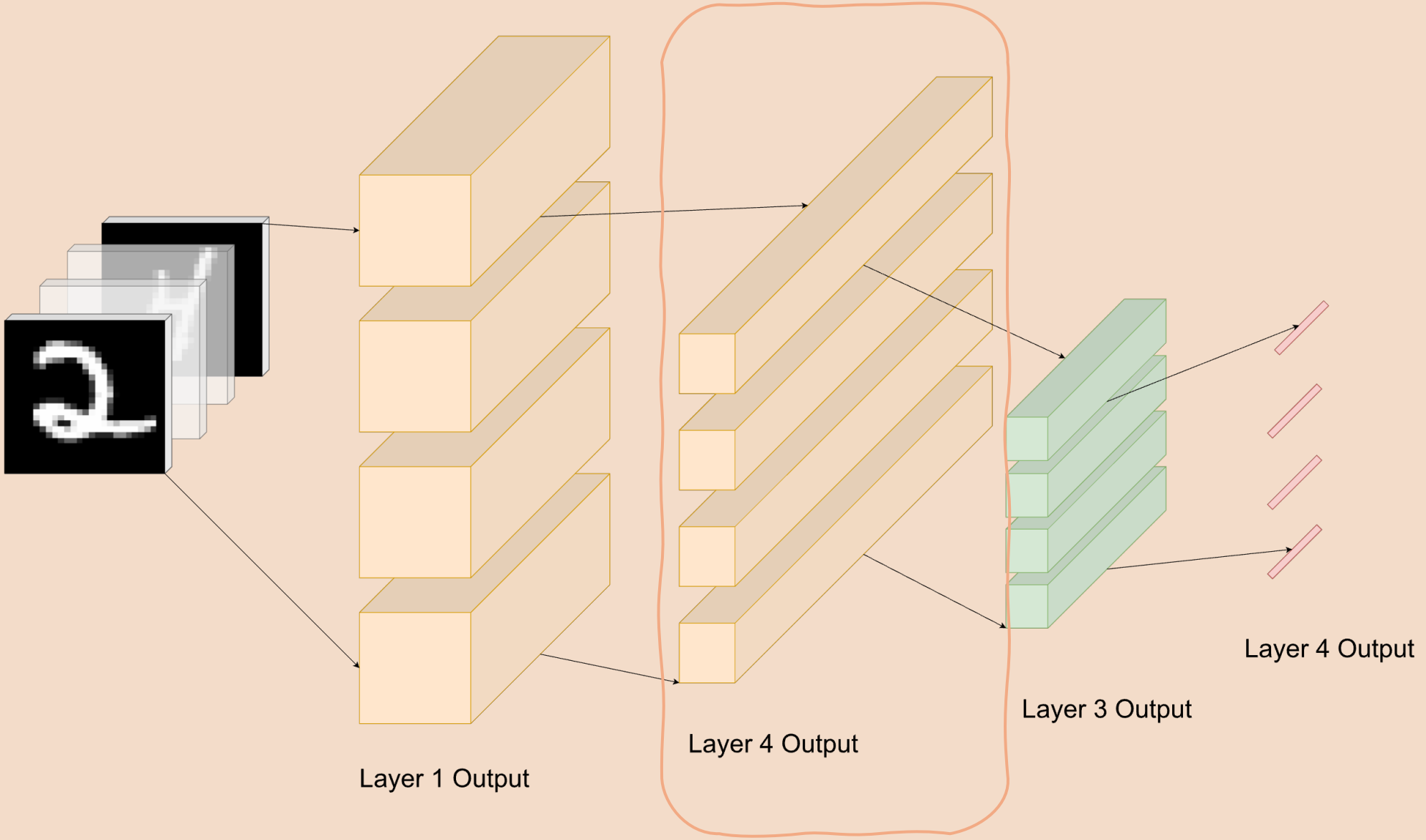


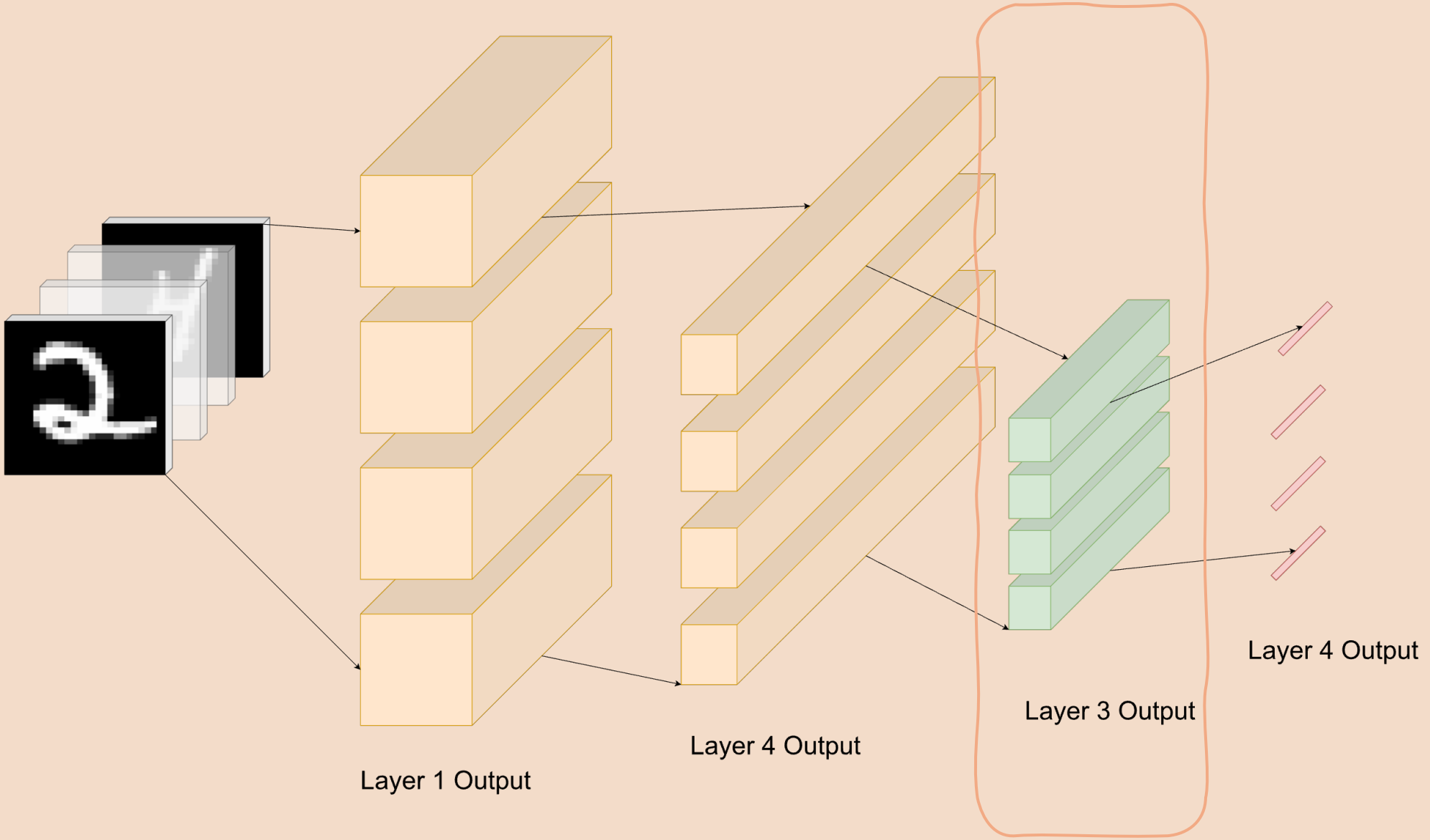
Chapel Parallelism

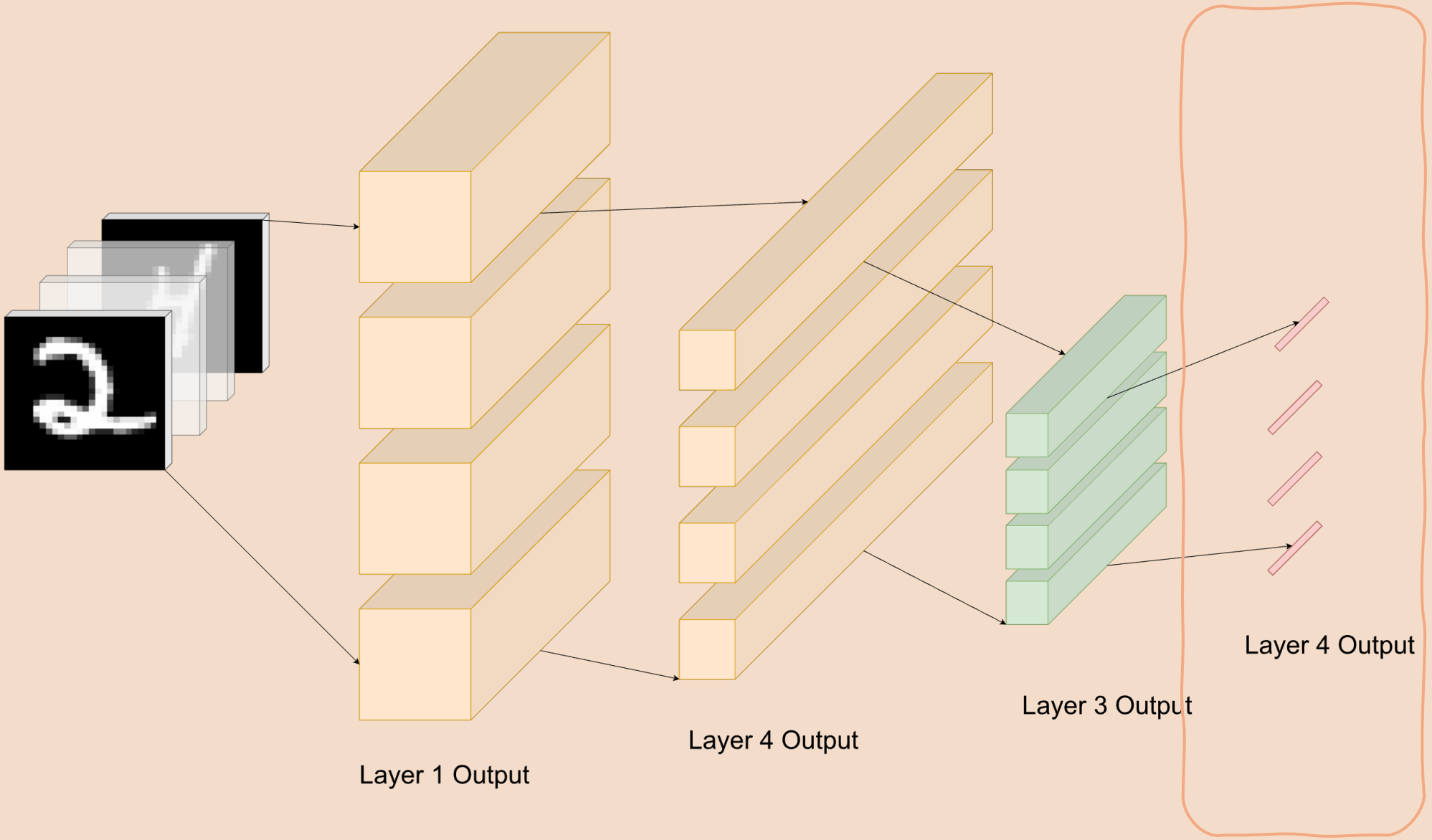


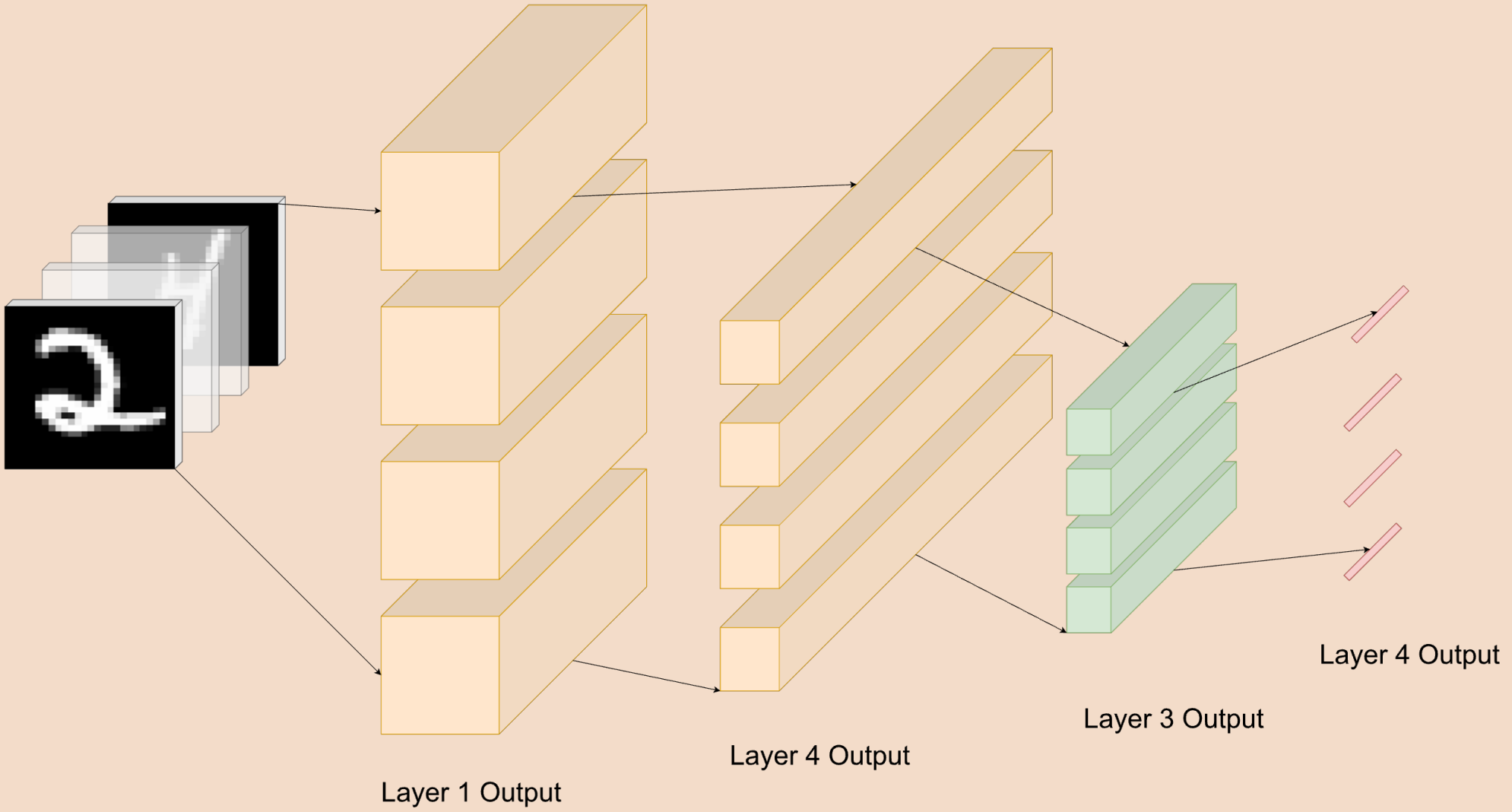

Layer 1 Output
Layer 4 Output
Layer 3 Output
Layer 4 Output


Layer 1 Output
Layer 4 Output
Layer 3 Output
Layer 4 Output


Layer 1 Output
Layer 4 Output
Layer 3 Output
Layer 4 Output


Layer 1 Output
Layer 4 Output
Layer 3 Output
Layer 4 Output


Layer 1 Output
Layer 4 Output
Layer 3 Output
Layer 4 Output


Layer 1 Output
Layer 4 Output
Layer 3 Output
Layer 4 Output

Domain Ergonomics

Domain Ergonomics

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

Domain Ergonomics

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

Domain Ergonomics

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

Domain Ergonomics

```
tensor(n,type t)
```

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

Domain Ergonomics

```
tensor(n,type t) ≅  [dom] t where dom.rank == n
```

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

Domain Ergonomics

```
tensor(n,type t) ≅  [dom] t where dom.rank == n

type Tensor(n) = tensor(n,real);
```

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

Domain Ergonomics

```
tensor(n,type t) ≅  [dom] t where dom.rank == n

type Tensor(n) = tensor(n,real);
```

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: Tensor(1)) { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

Domain Ergonomics

```
tensor(n,type t) ≅  [dom] t where dom.rank == n

type Tensor(n) = tensor(n,real);
```

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: Tensor(1)) { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc arr_f(a: Tensor(1)): Tensor(1) { ... }
```

```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

Domain Ergonomics

```
tensor(n,type t) ≅ [dom] t where dom.rank == n

type Tensor(n) = tensor(n,real);
```

```
proc arr_f(a: [] real)
  where a.rank == 1 { ... }
```

```
proc arr_f(a: Tensor(1)) { ... }
```

```
proc arr_f(a: [] real): [] real
  where a.rank == 1 { ... }
```

```
proc arr_f(a: Tensor(1)): Tensor(1) { ... }
```

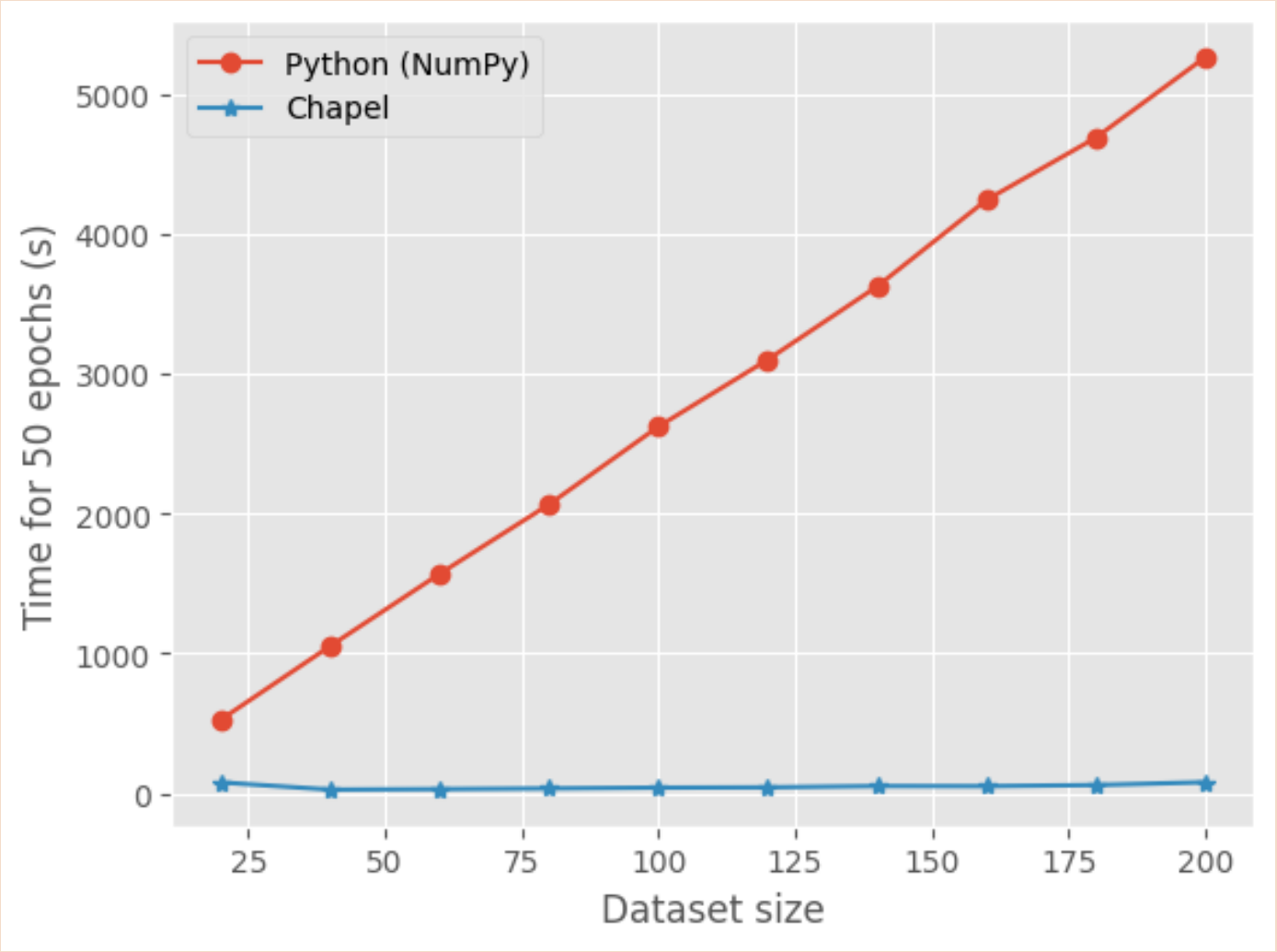
```
proc conv(ker: [] real,
          imgs: [] real): [] real
  where ker.rank == 2 && imgs.rank == 3
    { ... }
```

```
proc conv(ker: Tensor(2),
                imgs: Tensor(3)): Tensor(3) { ... }
```

Parallelism Abstractions

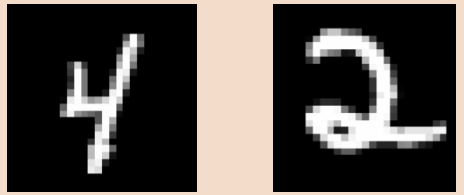
```
proc f(x: real): real { return x + 1; }

var a = // large array
var b = f(a);
// or
var b = a + 1;
```

Performance

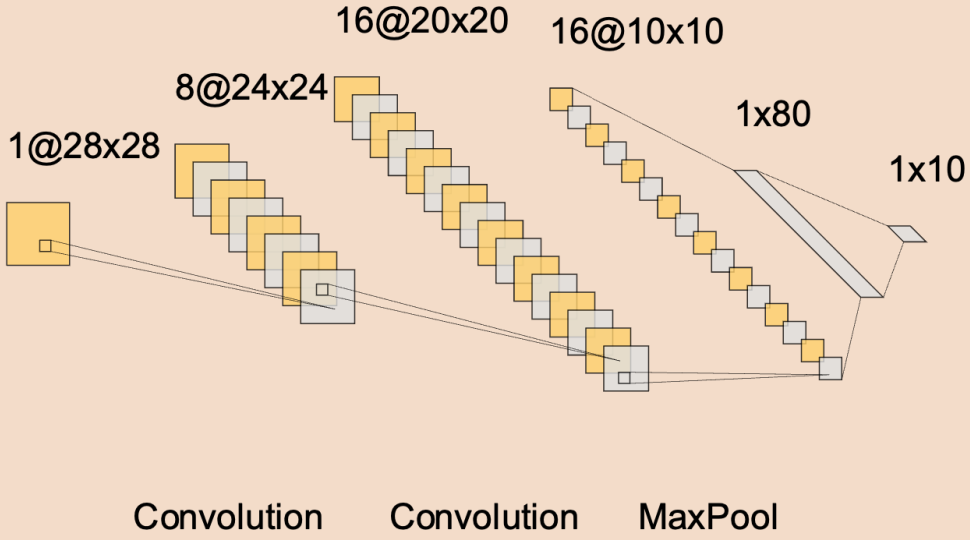


*Maybe with OpenBLAS 64 backend for NumPy

Input:



Output: 0,1,2,3,...,9



More work to be done in 2024

Training

Necessitates automatic differentiation

Should feel similar to PyTorch or TF

Inference

Sufficiently large toolkit for tensor manipulation operations

Interface with Arkouda

Adaptable to any HPE computing solutions (leveraging CUDA, multi-node distribution)