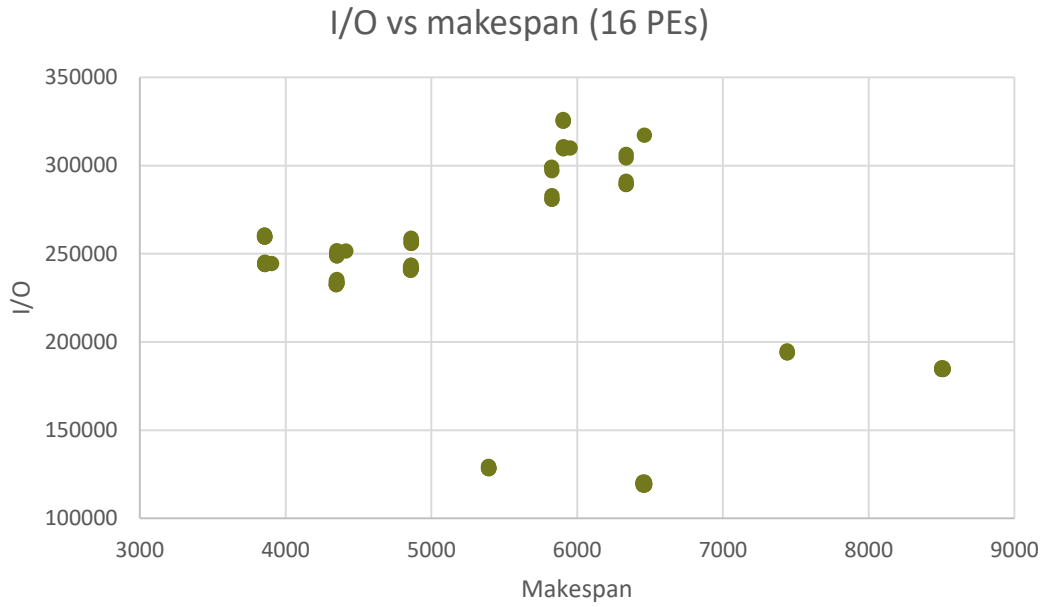
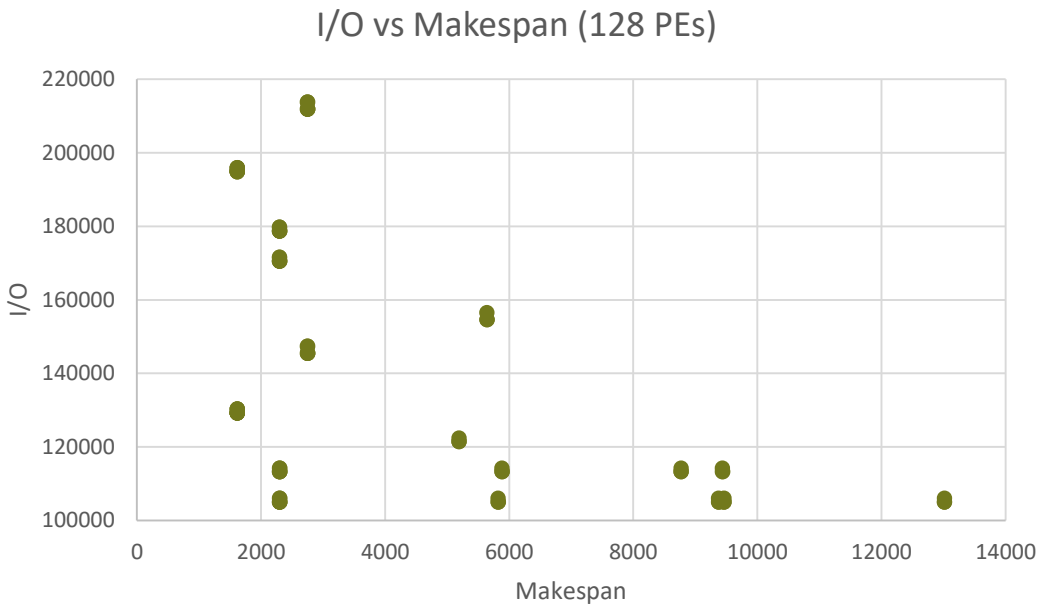
ASA: Enabling DSE

ASE

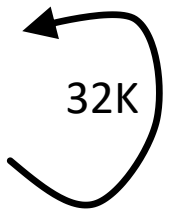
Performed Application Space Exploration + Scheduling. There are 4 MMMs, this results in 81 appl. variations

8 PEs		16 PEs	
Makespan	I/O	Makespan	I/O
6735	269274	3856	260442
6735	268634	3856	259418
6735	269402	3856	260570
6725	268762		
	261338		
	209264		
	209264		
	219376		
	218480		
	218480		





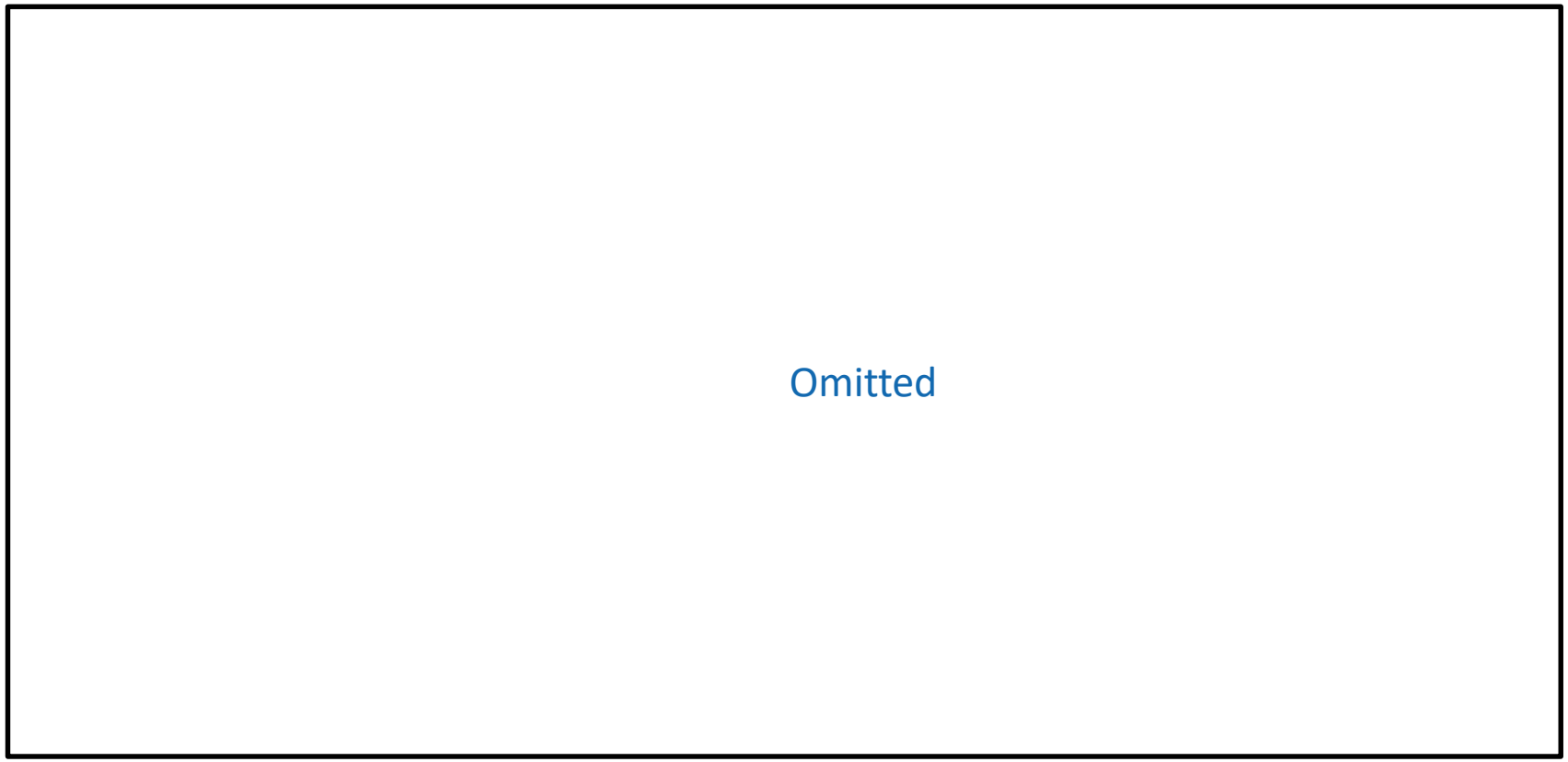
Iterated computation



How to represent and analyze this conveniently?

Iterative computation – Fully unrolled approach

Explicitly (fully unrolled), both in the SDFG and in the Canonical DAG



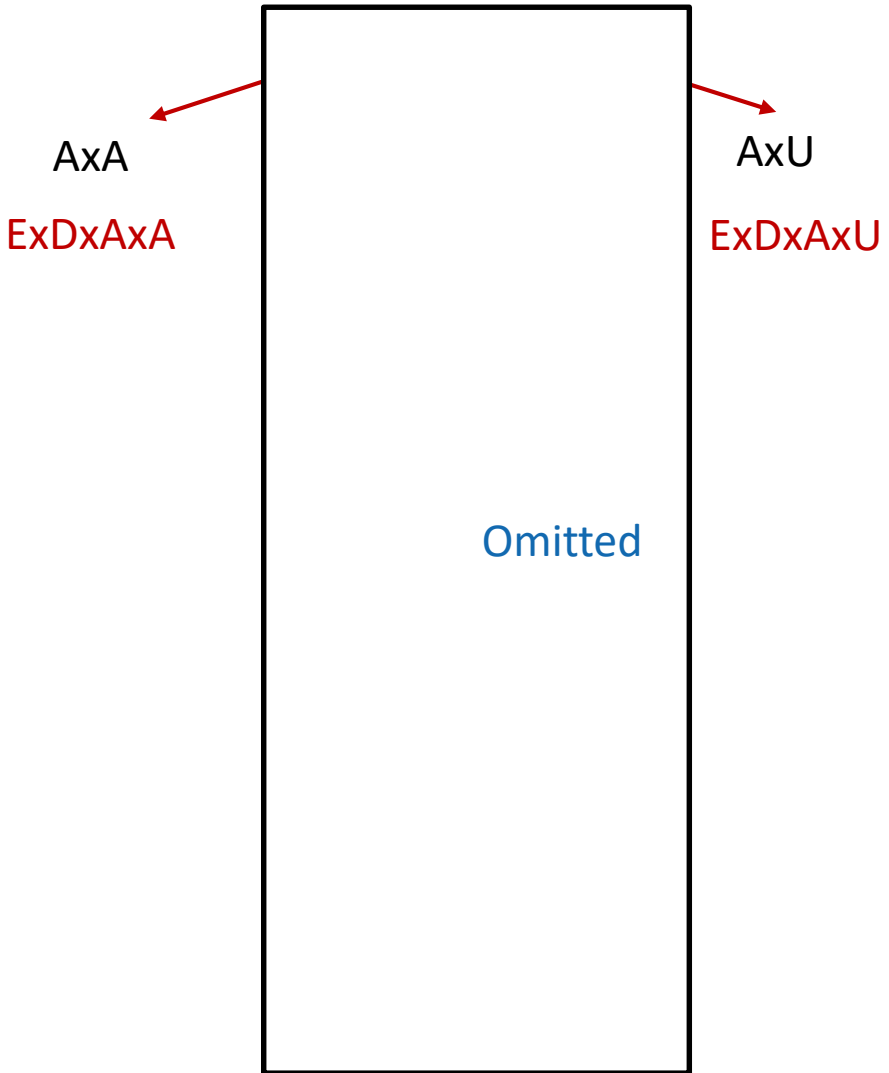
Pro: Would allow capturing parallelism across independent iterations

Cons:

- For embar. parallel computations, we don't want to have a different representation of the same operation in each iteration (this will make the Space Exploration explode)
- Too expensive: SDFG with millions of nodes, DAG with 100Ks nodes

Iterative computation – Tensor approach

Instead of considering single matrices or vectors as input, consider 4D or 3D tensors



Similarly, also the operation implementations must change, moving to batched operations or tensor operations (when applicable)

Pro: More concise and quick to evaluate

Cons: But will limit the parallelism (difficult to exploit among multiple iterations)

Iterative Computation – Partial Unrolling

Represent the iterative computation in DaCe, then partially unroll it in the Canonical DAG

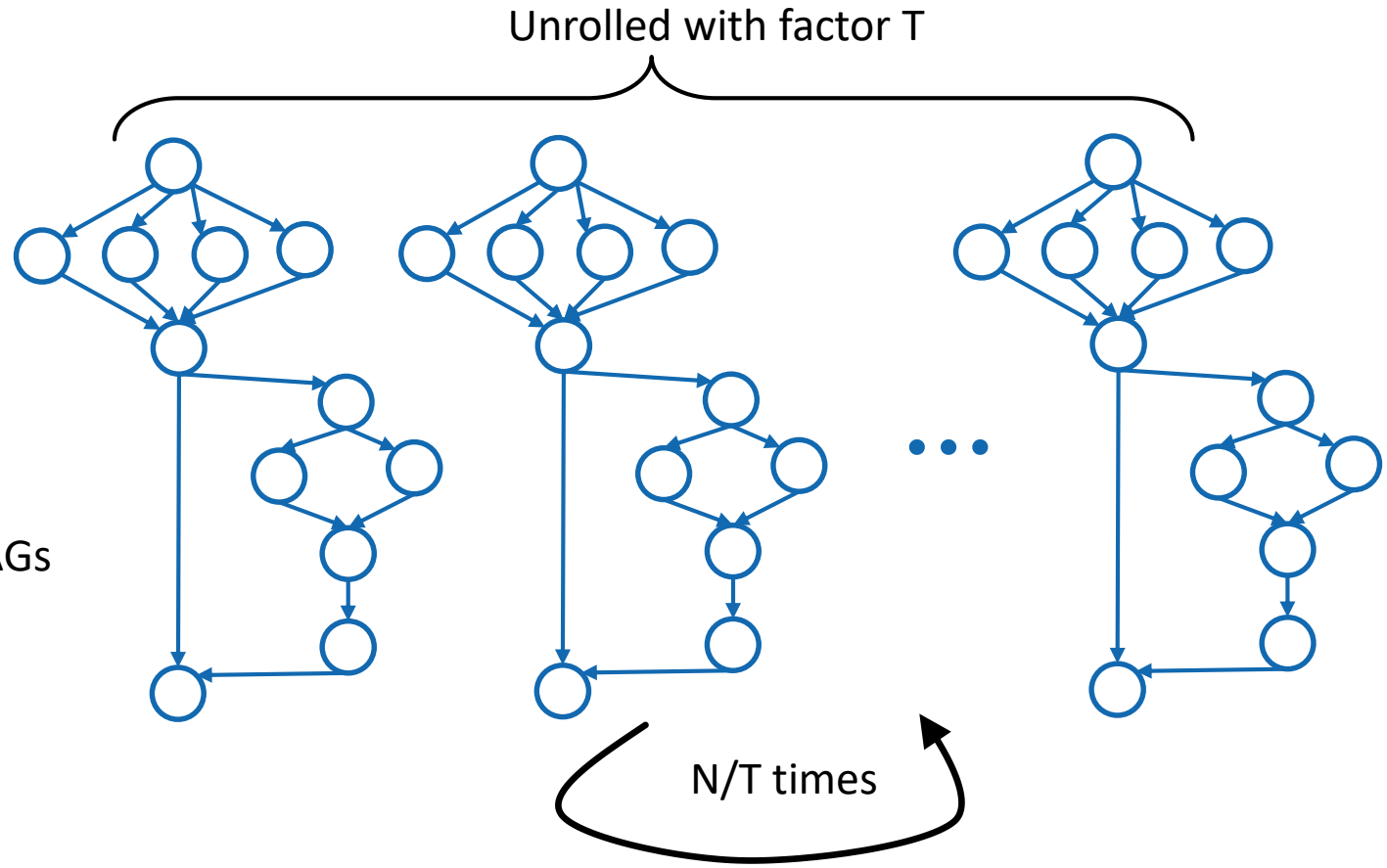
ExDxAxU

But, internally, we use single matrices

Map representing independent iterations

ExDxAxA

Canonical DAGs



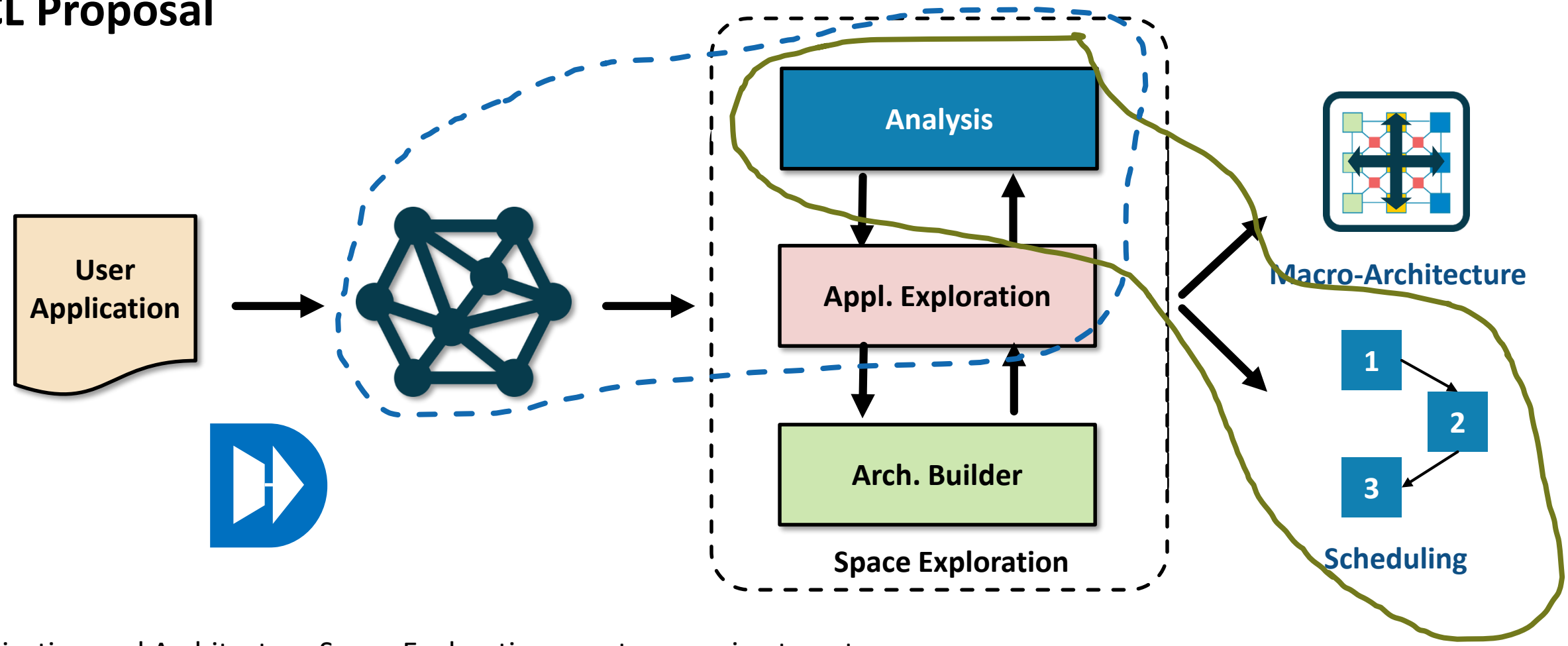
Pro:

- Concise/cheap on the SDFG side
- Will allow us to capture inter-iteration parallelism

Cons: Unrolling factor is another dimension to explore (but we can keep it under control)

Planning

FCL Proposal



Application and Architecture Space Explorations are two moving targets

Next big milestones:

- March: discuss the 5G case study
- End of the project: ML workloads

Actually they are both carried out in parallel (Breadth First search instead of Depth First Search)

Things to do (not in chrono order will be done, some optional right now)

Use cases: be able to represent applications through DaCe and with Canonical DAG:

- 5G use case: be more compliant with orig. appl
- ML: something more interesting than Lenet: e.g., Resnet, Encoder.

Explore Application representations: be able to "compile" the application to Canonical DAGs

- Conveniently represent iterative algorithms (see discussion of today)
- Support considered use cases

Explore Architectures: be able to "consider" different macro-architectures

- We can change the number of PEs, this will affect the scheduling of the Canonical DAG
- Changing the PEs (but still under the homogenous PE assumption)
  - Support certain type of operations? (this affect the SDFG expansions)
  - [Vectorization]

Things to do (not necessarily will be done)

Space Exploration Goals and Optimization:

- Goals: optimize/minimize performance/power/area: we need way of estimating these
  - Performance is given by the scheduling makespan
  - Area: # of PEs, but also on-chip buffer space (e.g. for deadlock prevention)
  - Power: directly proportional to the off-chip memory accesses
- [Pruning the search space]

Publication plan: all of this needs to be conveyed in a publication(s). Venue TBD

I would need your help for:

- Expressing your application in DaCe: the publishable part. Have a full, baseline, version, with my support
- PPA models