My View on the Path to Artificial General Intelligence

Fei Sun

Compute Technology Lab, DAMO Academy
May, 2021

Disclaimer

- This is NOT a scientific presentation
- This presentation contains many
  - Hypothesises
  - Projections
  - Extrapolations

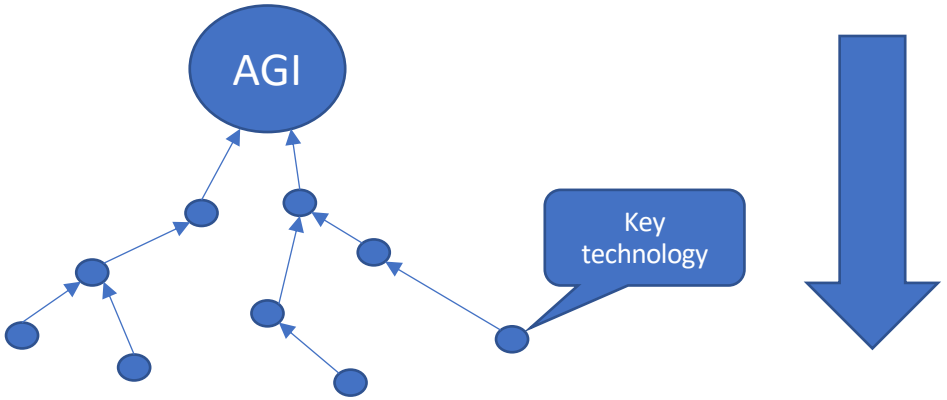
Contents in ATA

- 自上而下思考实现AGI技术难点及可能方法
  - https://topic.atatech.org/articles/204693
- 稀疏是通往AGI的必由之路
  - https://topic.atatech.org/articles/204694
- AGI，从我做起
  - https://topic.atatech.org/articles/204695

The Approach

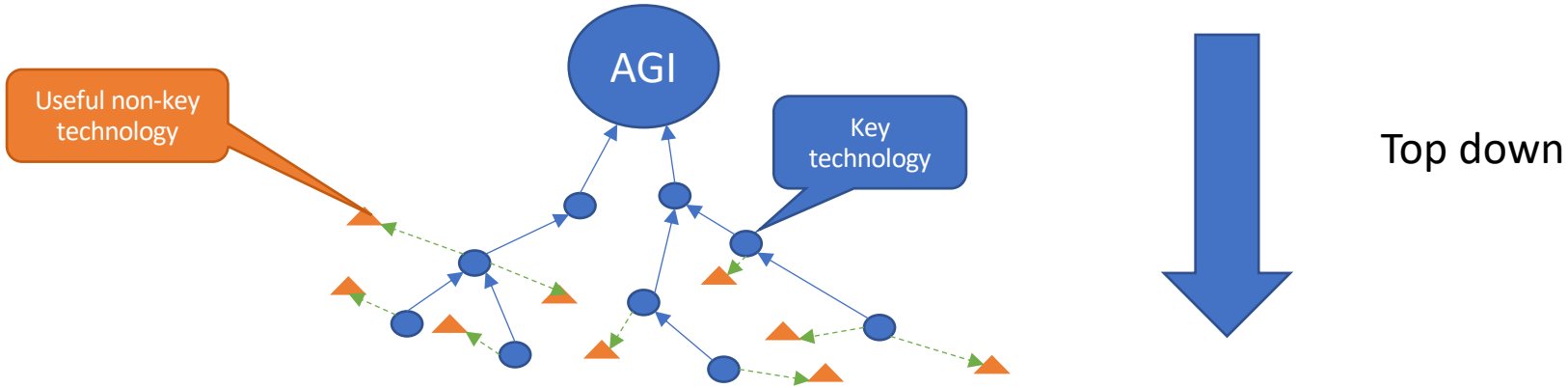
Path towards AGI

Path towards AGI

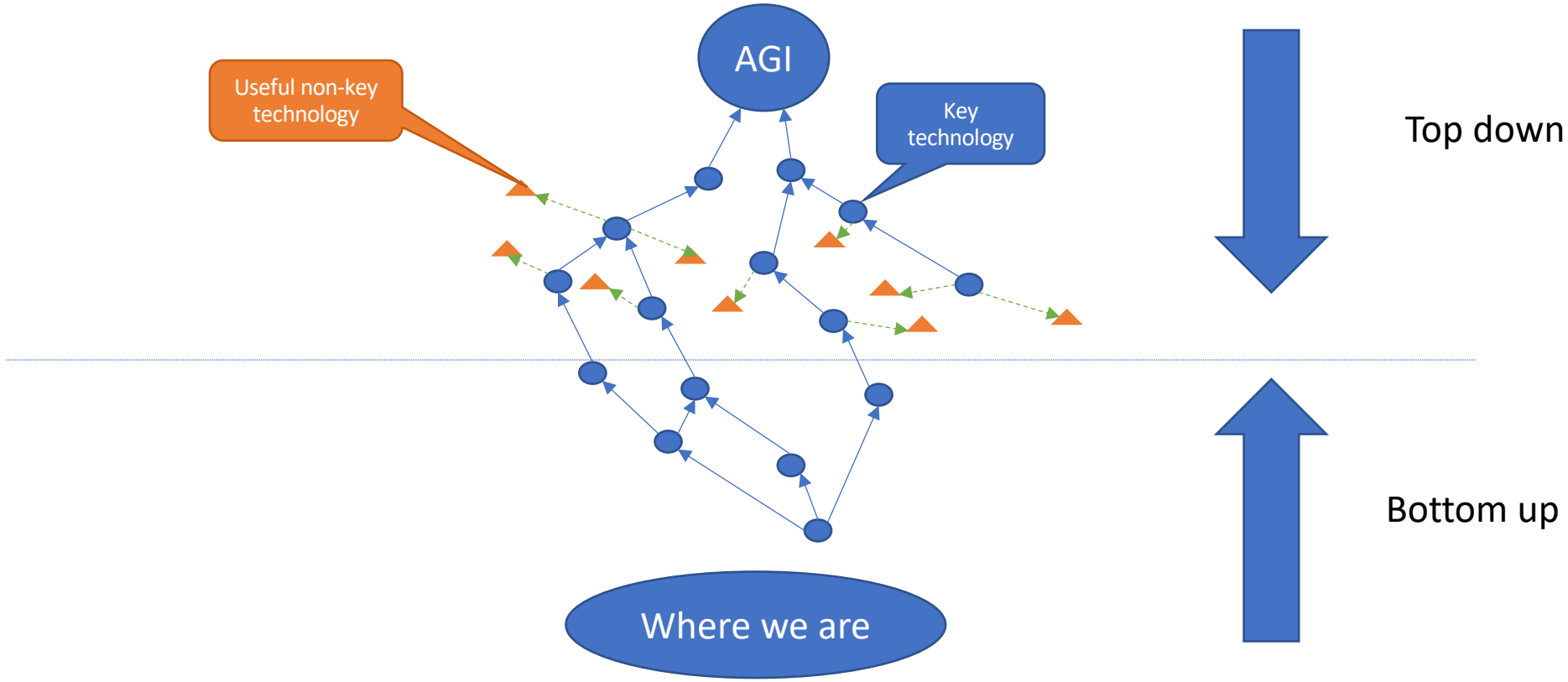


Top down

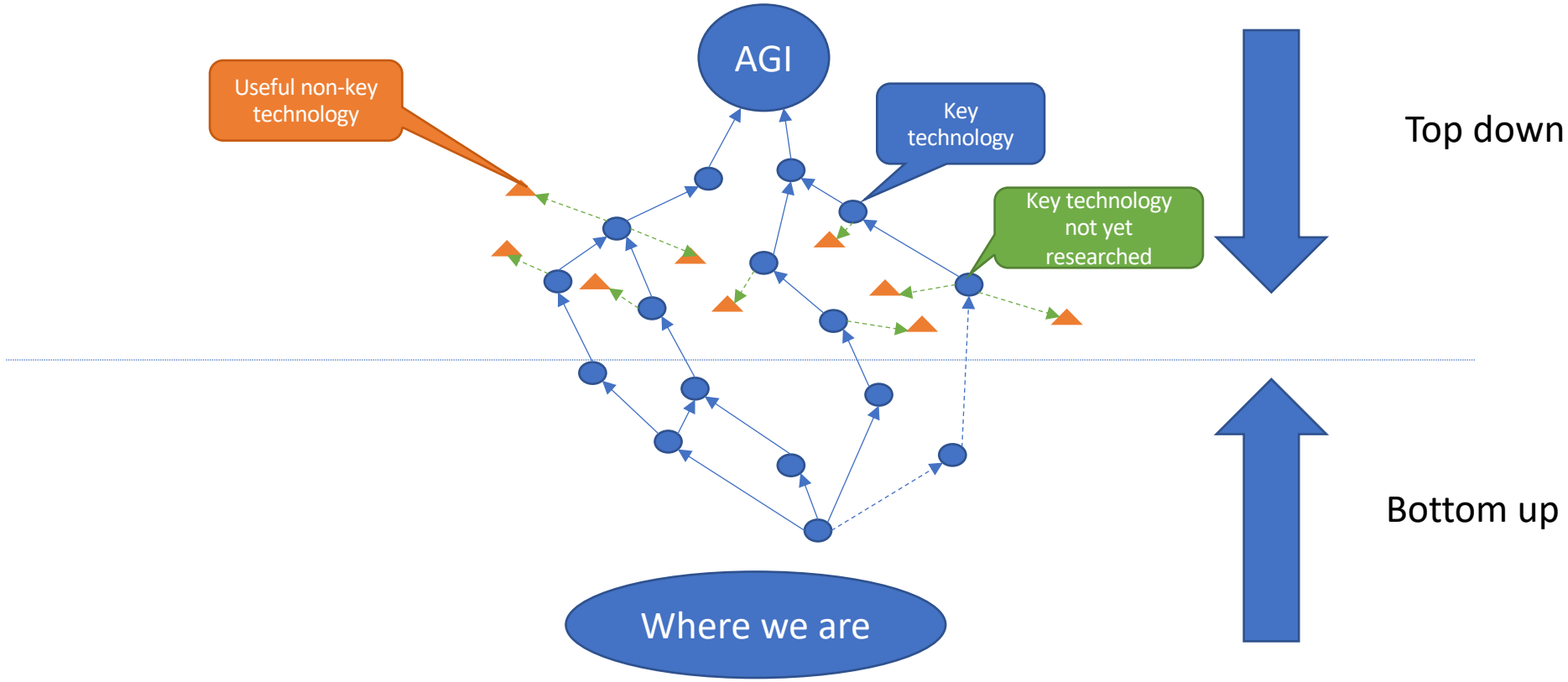
Path towards AGI



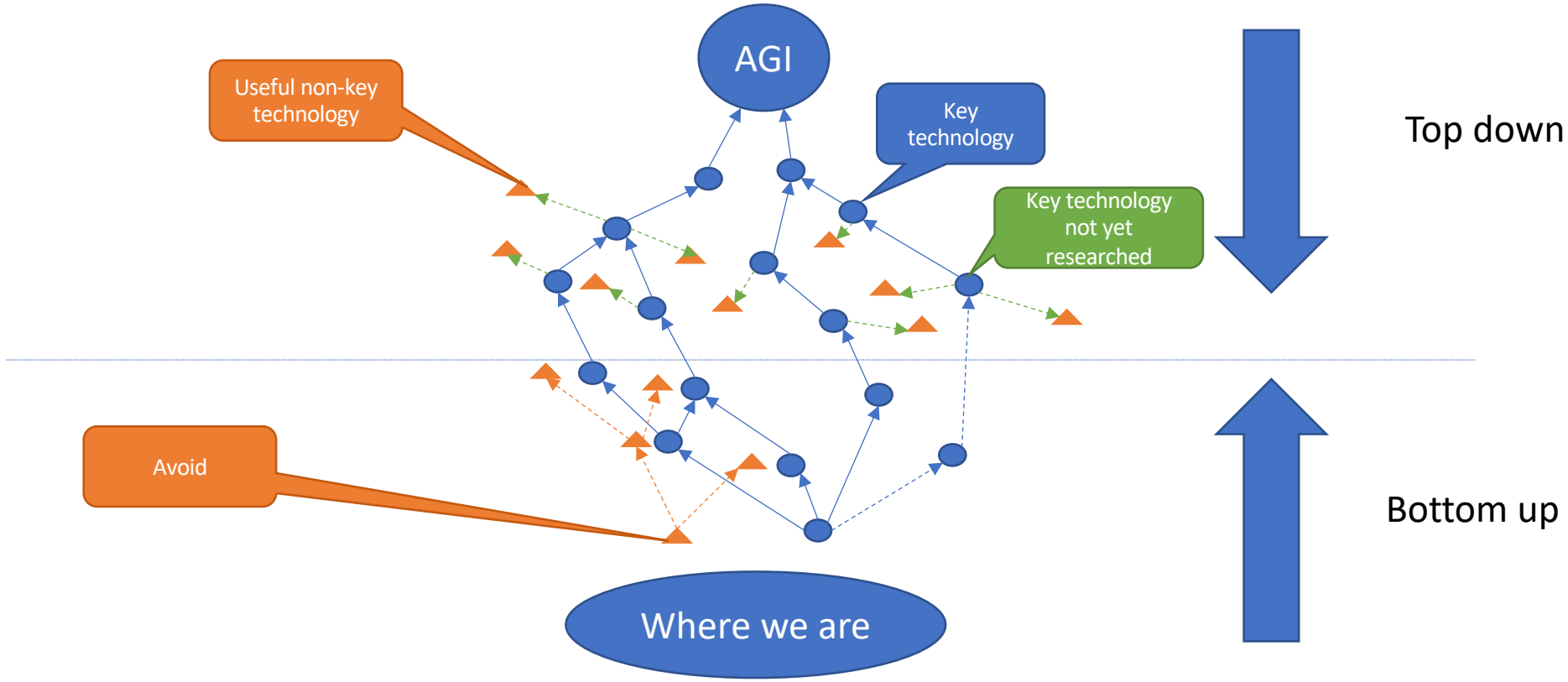
Path towards AGI



Path towards AGI



Path towards AGI



Outline

- Top down view of the path breakdown towards AGI
  - The ability to learn
  - The ability to evolve
  - The ability to scale
  - Composition
- Sparsity is an essential step towards AGI
  - Why sparsity is important
  - Sparsity on ultra large models
  - Sparsity on memory, composition, and evolution
- AGI, starting from myself
  - People’s view on AGI
  - AGI in academia and industry
  - Interesting near term projects

Top Down View of Path Breakdown Towards AGI

Preamble

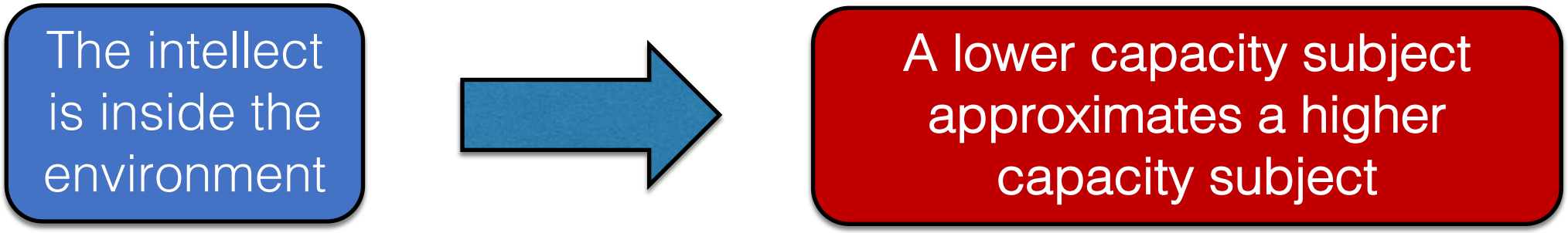
- Some contents are taken from my previous writings in 2017:
  - A few predictions on artificial intelligence[18]
  - Some scribble of things[19]

What is the goal of an intellect?

- Goal: survive
- Methodology: approximate the environment, the past, the present, and the future.

What is the goal of an intellect?

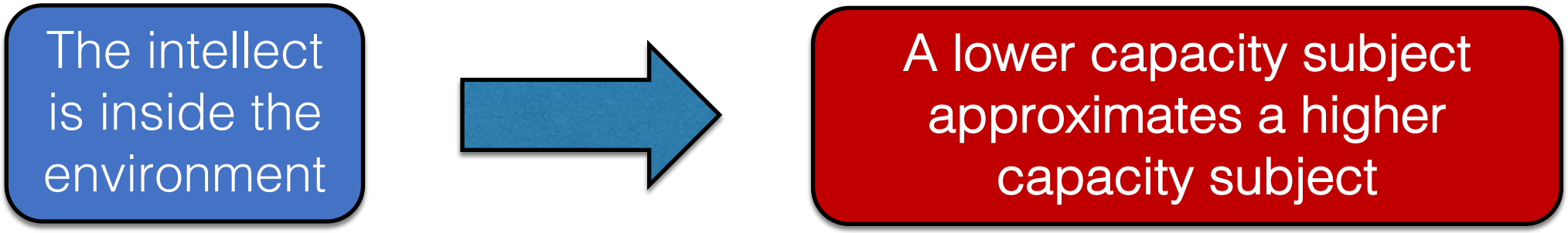
- Goal: survive
- Methodology: approximate the environment, the past, the present, and the future.



* Some scribble of things[19]

Over-parameterization

- Locally over-parameterize, globally under-parameterize



* Some scribble of things[19]

Three abilities for AGI

The ability to learn

- The skills an agent acquires in its lifetime
  - Perception
  - Memory
  - Reasoning

The ability to evolve

- The skills an agent passes from one generation to the next
  - Embedding
  - Factory

Composition

The ability to scale

- The required external help/cost of passing the skills
  - Below a self-sustaining threshold

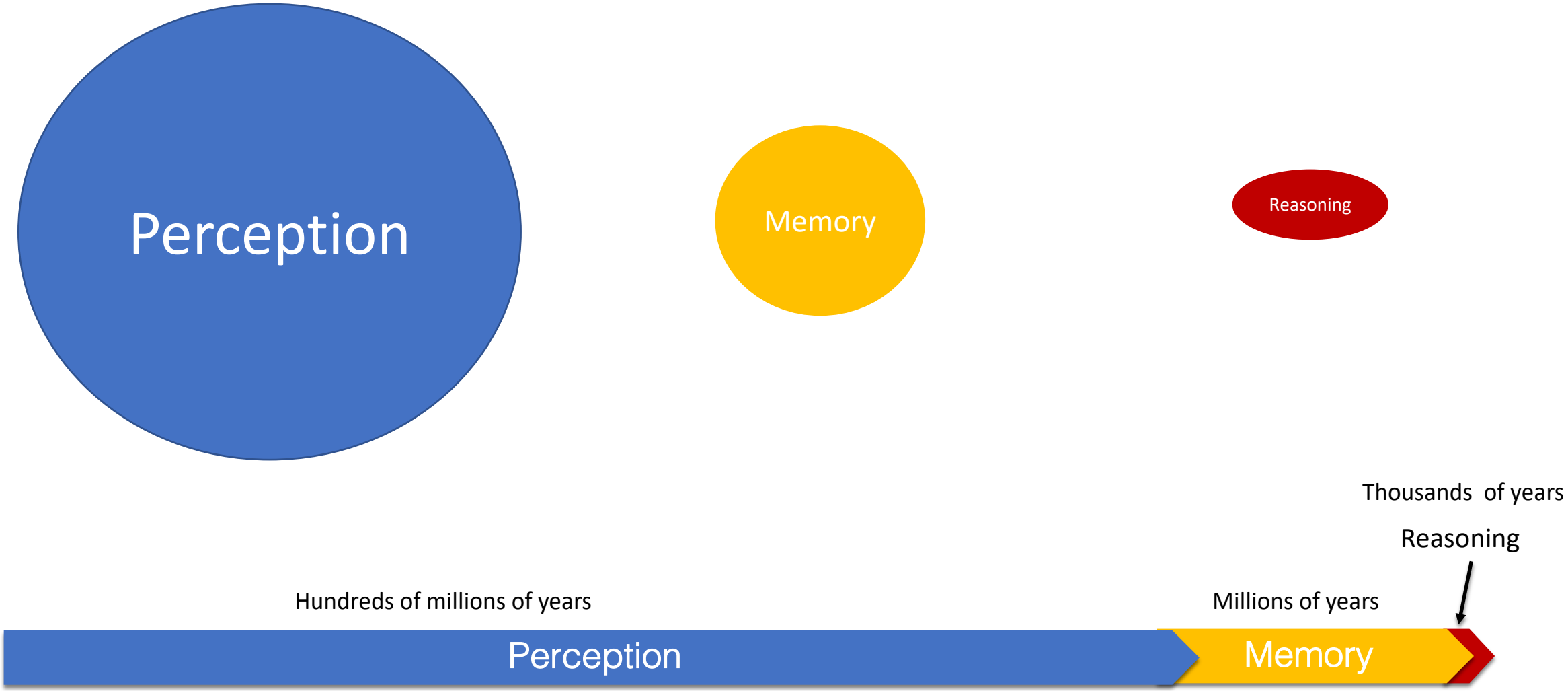
* Hypothesis: AGI cannot be achieved via pure learning

The ability to learn

The ability to learn

- An agent’s internal structure is (almost) fixed
  - Learning capability is limited by the structure
- Hypothesis: learning is more computation efficient than evolving
  - Likely be (variations of) SGD based approaches

Three components of learning



Perception

- “More than half the brain is devoted to processing sensory information”[1]
- Transfer spatial coordinate systems with dimension reduction
- Feature extractor
- Majority of DNN research is focused in this area
  - Higher capacity
    - Larger models
    - Higher abstraction level -> More efficient feature extractor (structure)

Memory

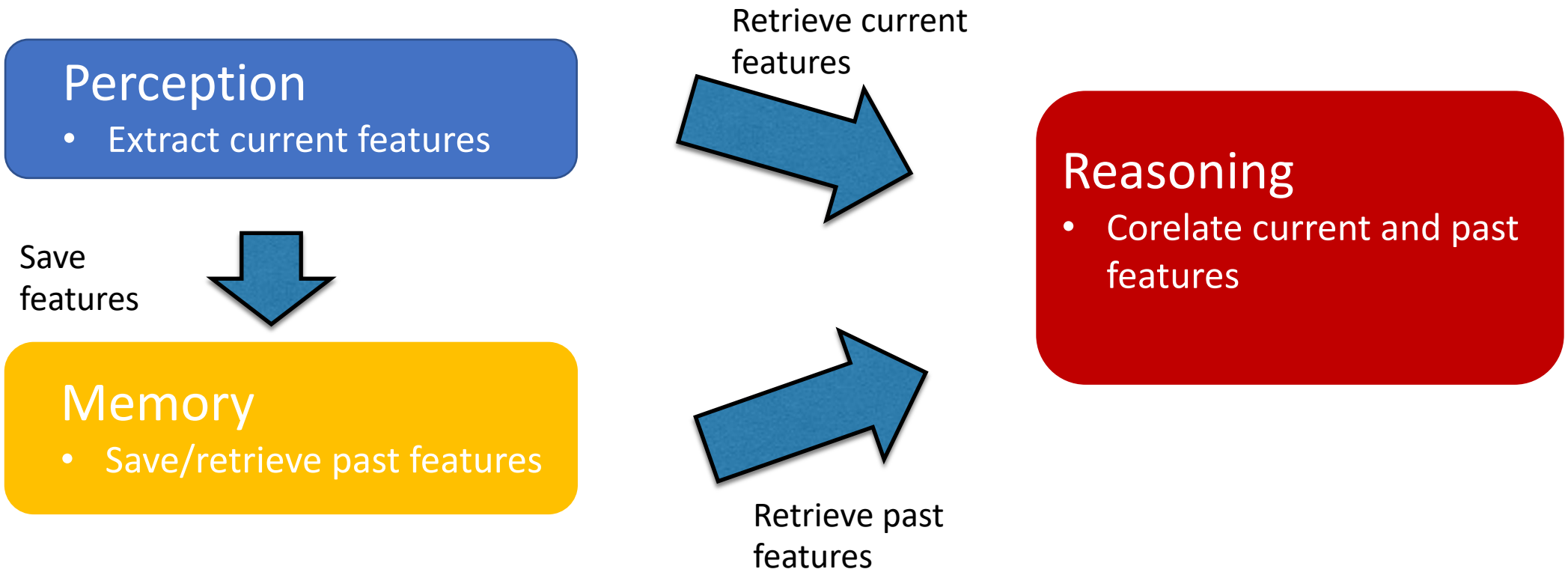
- Transfer temporal coordinate system
- Extract temporal features

- Currently not well studied
  - Memory embedded in the feature extractors (RNN, MLP)
  - Neural Turing machine[7]
  - Differentiable neural computer[8]

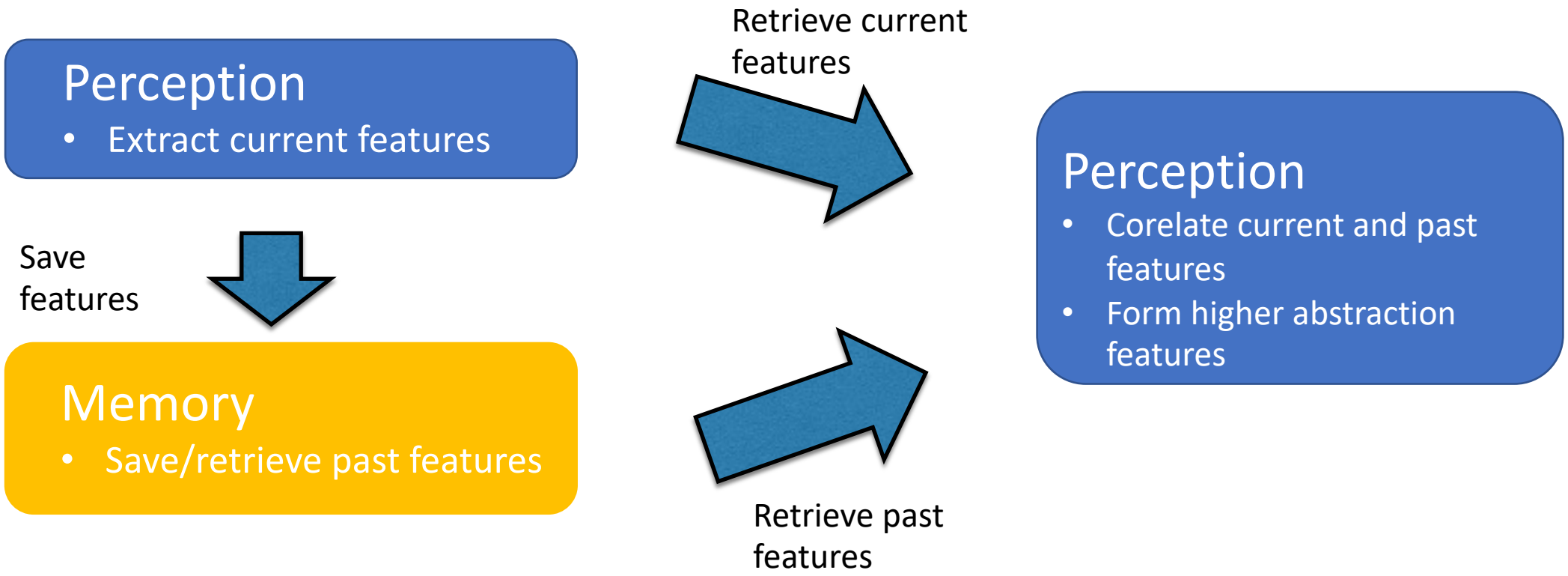
Reasoning

- Reasoning is the process that corelates the spatial features (perception) and temporal features (memory), and predicts future features
  - No clear cut boundary with high level features
- Low level reasoning is baked in the model and cannot be separated out
  - Inductive reasoning, analogical reasoning
- Higher level reasoning is not explored
- Hypothesis: reasoning is not difficult
- Hypothesis: composition is key to reasoning

Perception, memory, reasoning



Perception, memory, reasoning



Perception, memory, reasoning

- Existing approach
  - Monolithically mixing perception, memory, reasoning
    - OK for short term memory, low level reasoning
    - Not OK for long term memory, high level reasoning
    - Not scalable

The ability to evolve

The ability to evolve

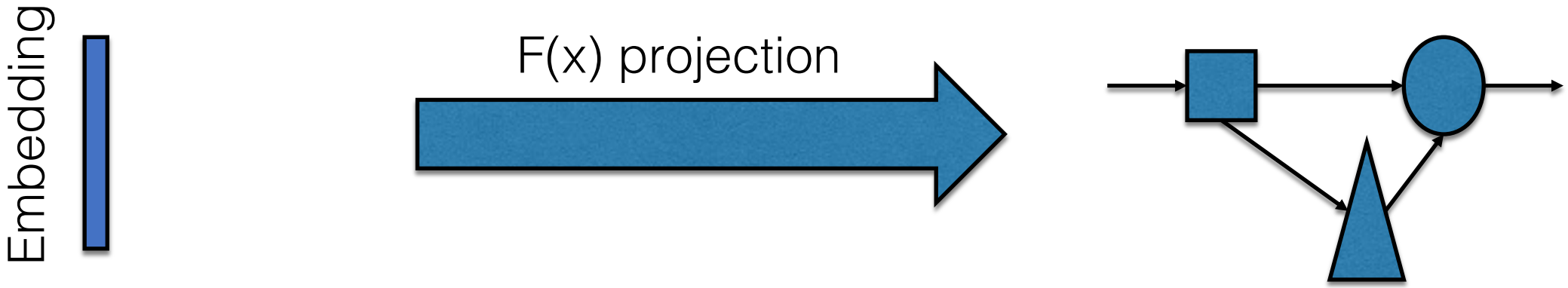
- An agent's structure is significantly changed to increase learning capability
  - Usually require many agents work together
- Hypothesis: structure change is more difficult than learning
  - Require a lot more computation
  - Cannot rely on SGD based approaches
- Difference from learning
  - Learning is SGD based, evolution is not.

Current status

- Manual design structure by humans
  - ResNet, EfficientNet, LSTM, Transformer etc.
  - Not evolution
- Neural architecture search
  - Search structures directly ->too many variables
  - Building blocks specified by humans, rules specified by humans -> inflexible

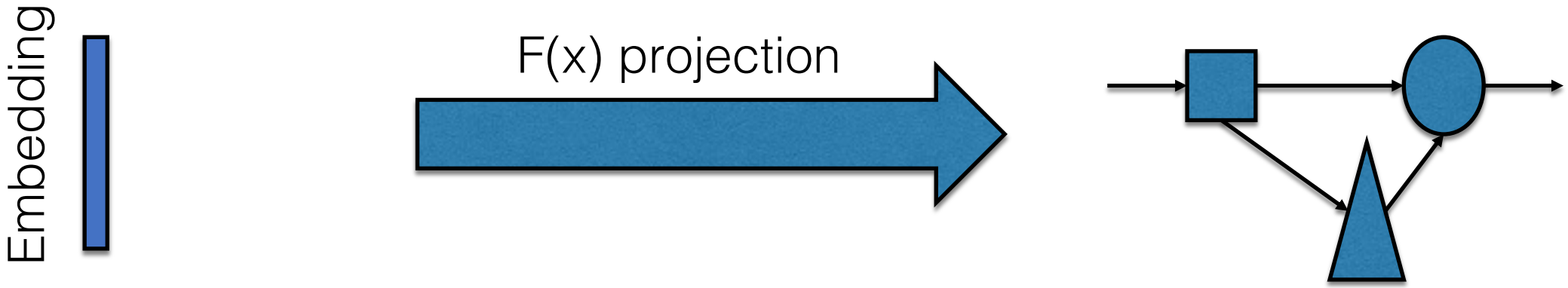
How to reduce evolution complexity? - Embedding

- Search structures directly ->too many variables
  - Embed the structure with fewer variables



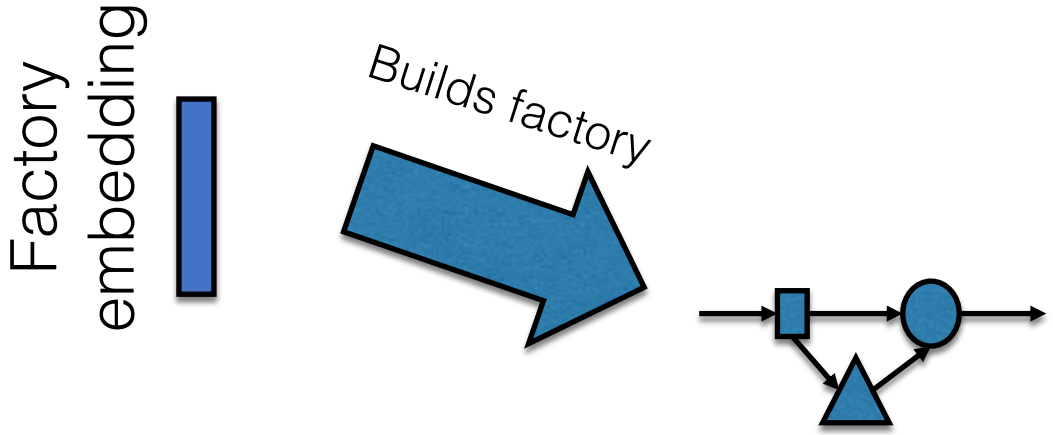
Embedding

- Learning the embedding in training
  - Less variation
- Evolution is easier
  - Evolutionary algorithm: mutation, crossover on encodings



How to increase evolution variation? – Factory*

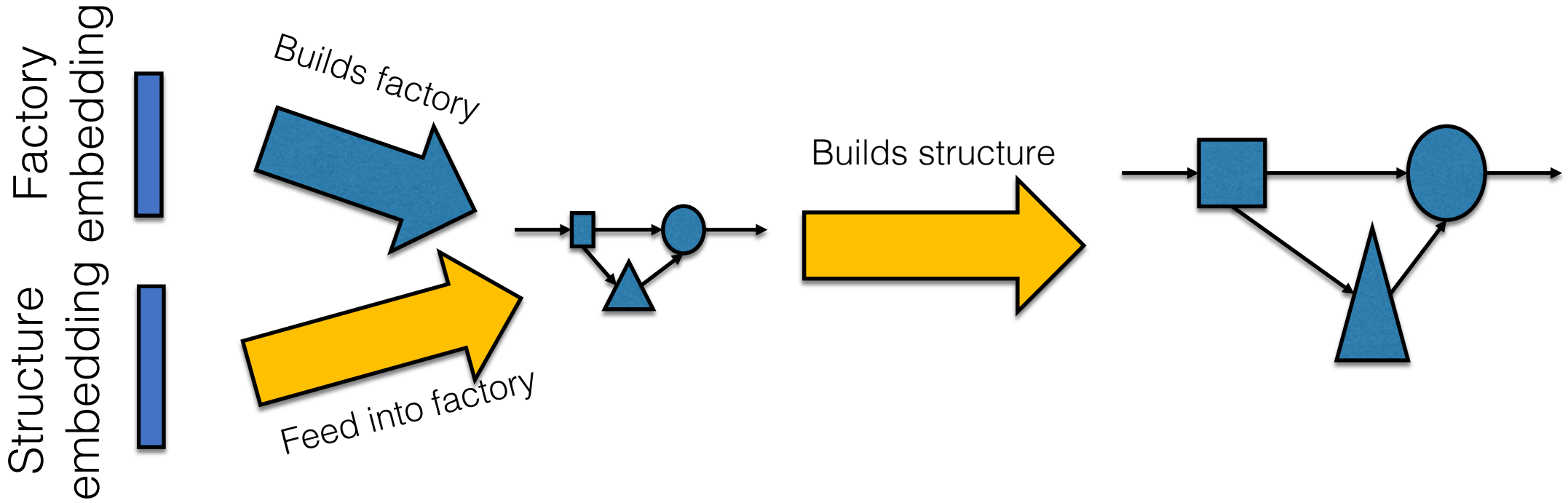
- Building blocks specified by humans, rules specified by humans -> inflexible
- Rules are specified in embedding



* Name borrowed from “Design patterns: elements of reusable object-oriented software”[2]

Factory

- First builds factory from factory embedding
- Then feeds structure embedding to factory to build structure
- Factory can be hierarchical



The ability to scale

The ability to scale

- Learn: single agent with fixed structure
- Evolve: one or many agents upgrading structures
- Scale: the cost of upgrading structure is less than the reward from the upgraded structure
  - Massive population enable rapid evolution
  - Positive feedback
  - Reaching critical point

Current status

- Not yet explored
- Long term future research direction when near term research objectives are achieved

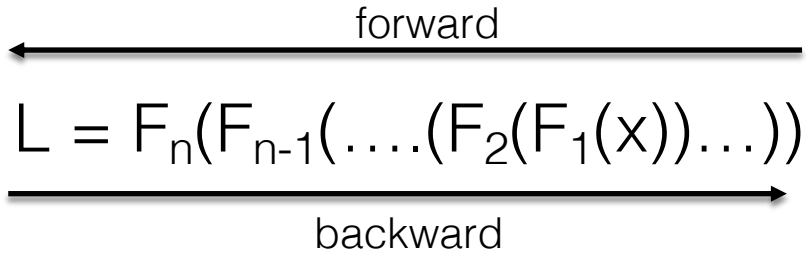
Composition

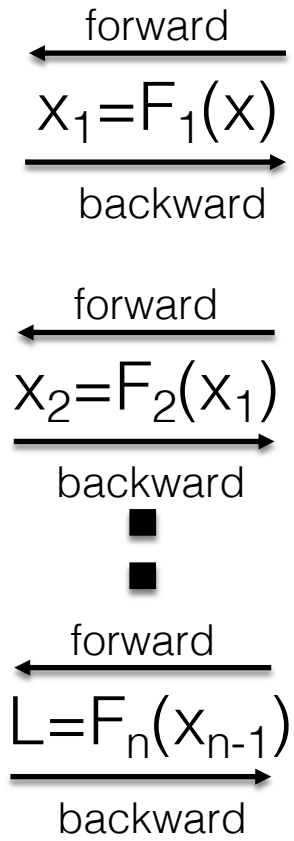
Function Composition

- y = f(g(x)) -> reuse f, g
- DNN models are function compositions
  - $L = F_n(F_{n-1}(….(F_2(F_1(x))…))$
- Can training be done in a function composition way?

Function Composition

- Can training be done in a function composition way?





With the goal of reusing $F_1, F_2, \ldots, F_n$?

i.e. can we train $F_1, F_2, \ldots, F_n$ separately?

Note $F_1, F_2, \ldots, F_n$ are feature extractors

Object Composition

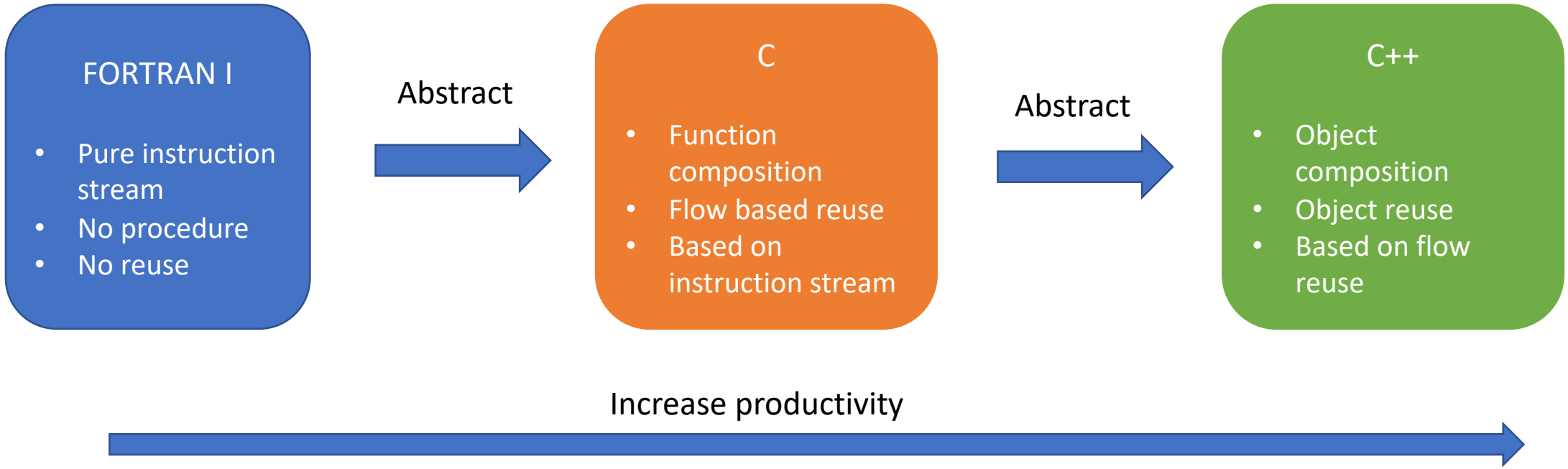
- Objects: a way to encapsulate data, internal procedures
  - -> internal relations are tighter than external relations
- Compound objects: different objects composed together to form a larger objects

Current status

- Monolithic, everything mixed together
  - Completely different approach from composition
- Pros:
  - Better quality (training can perform most fine-grained trade-offs)
- Cons:
  - Not scalable (the size of the model is limited, as the whole model needs to be trained)

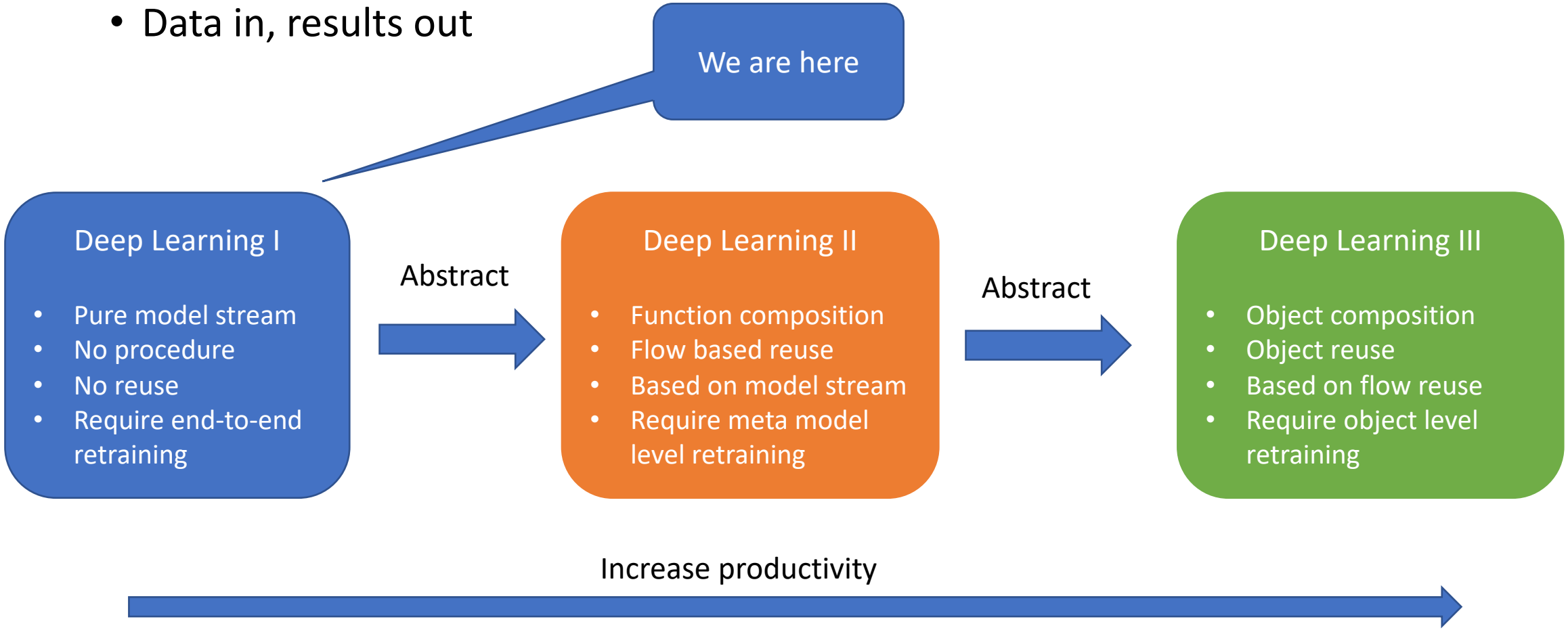
Software 1.0

- Humans write software programs



Software 2.0

- Deep learning is software 2.0[3]
  - Data in, results out



Deep learning, analog circuit, and digital circuit

Deep learning currently is more similar to analog circuit

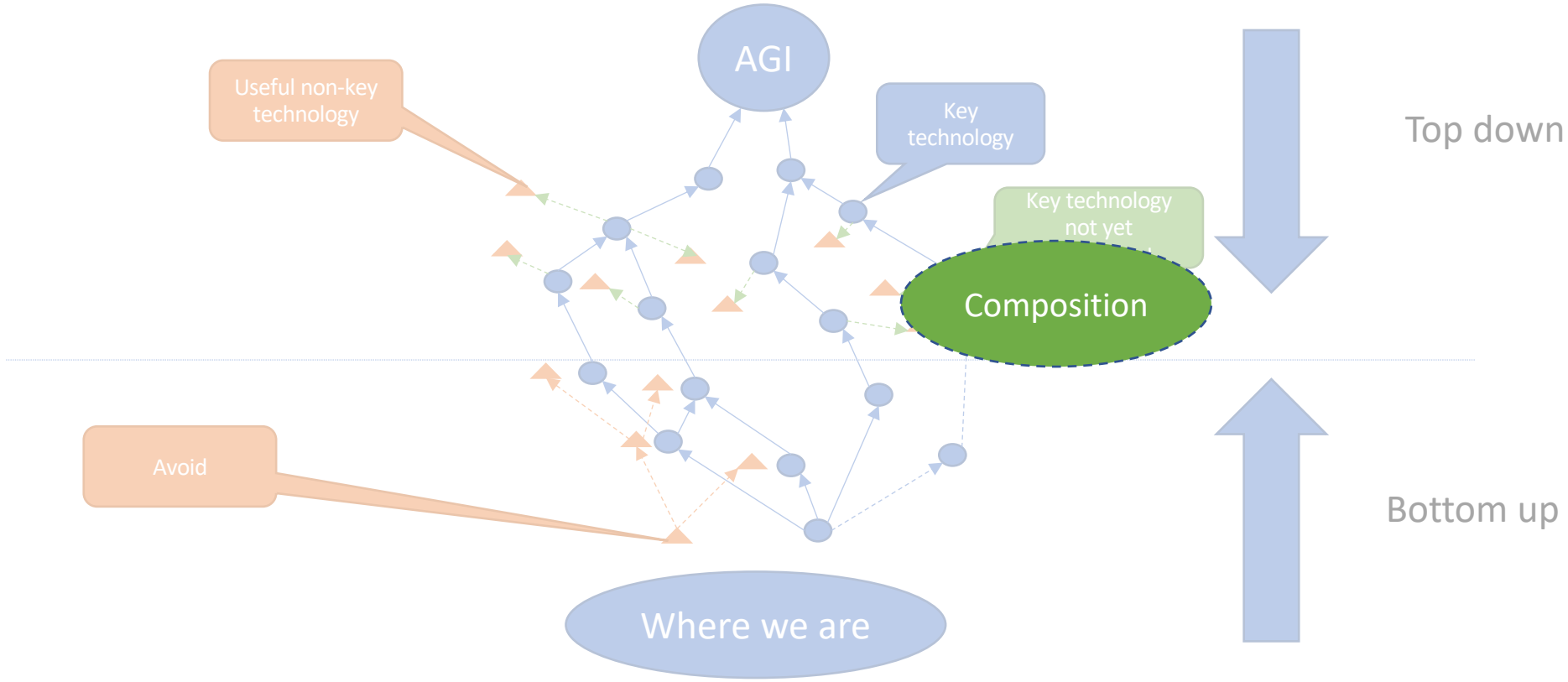
	Deep learning	Analog circuit	Digital circuit
Foundation	Back propagation and chain rule	Semiconductor physics	Boolean algebra
Impact of changes local or global?	Global	Global	Local
Can be composed?	No	No	Yes
Abstraction hierarchy	No	No	Device, circuit, gate, RTL, IP, SoC
Scalable?	No	No	Yes

Deep learning is desired to be more similar to digital circuit

Challenges

- The rule of composition
  - Can we find the mathematical foundation?
    - Similar to Boolean algebra to digital circuit
    - Different from finding the mathematical foundation for deep learning
- How to limit the implications for the existing DNN output?
  - Similar to digital circuit limits the implications of analog circuit
    - Is that possible? Analog circuit works on the linear region of MOSFET, while digital circuit works on the saturation region of MOSFET

Path towards AGI



Bottom Up Approach Identifying First Key Technology -- Sparsity

Outline

- Top down view of the path breakdown towards AGI
  - The ability to learn
  - The ability to evolve
  - The ability to scale
  - Composition
- Sparsity is an essential step towards AGI
  - Why sparsity is important
  - Sparsity on ultra large models
  - Sparsity on memory, composition, and evolution
- AGI, starting from myself
  - People's view on AGI
  - AGI in academia and industry
  - Interesting near term projects

What is dense?

- All weights and activations interact, forming fully connected network
- Fully connected MLP is dense
- All other computation are normalized to fully connected MLP

- Problems of dense
  - All weights and activations consume the same amount of computation
  - However, their contribution to the outcome varies a lot!

Sparsity definition IMO

- A mechanism to compute on the relevant features and weights in a model

Sparsity definition in a general sense

- A mechanism to compute on the relevant features and weights in a model
  - No mention of zeros
  - Zeros is a special case: zeros are always irrelevant
  - Dense is a special case: all features and weights are relevant

- Sparsity is not an absolute concept. It balances model quality and the amount of computation

Dense vs sparse

- Essence of dense
  - Sequential access of composing variables
- Essence of sparsity
  - Random access of composing variables

Is CNN a type of sparsity?

Is CNN a type of sparsity?

- Yes

- How do we compute it?

Is CNN a type of sparsity?

- Yes

- How do we compute it?
  - Using a fixed factory method

The choice of algorithms

- Problem: get the sum of every 5^{th} element in an 100-element array.

Algorithm 1:

```
sum=0;
for (i = 0; i < 100; i++) {
   sum += (i % 5 == 0) ? data[i] : 0;
}
```

Algorithm 2:

```
sum=0;
for (i = 0; i < 100; i+=5) {
   sum += data[i];
}
```

The choice of algorithms

- Problem: get the sum of every 5th element in an 100-element array.

Algorithm 1:

```
sum=0;
for (i = 0; i < 100; i++) {
   sum += (i % 5 == 0) ? data[i] : 0;
}
```

- Sequential access
- 5x accesses

Algorithm 2:

```
sum=0;
for (i = 0; i < 100; i+=5) {
   sum += data[i];
}
```

- Stride access
- No redundant accesses

The choice of algorithms, in deep learning

- Problem: get the sum of every 5th element in an 100-element array.

Algorithm 1:

```
sum=0;
for (i = 0; i < 100; i++) {
   sum += (i % 5 == 0) ? data[i] : 0;
}
```

- Sequential access
- 5x accesses

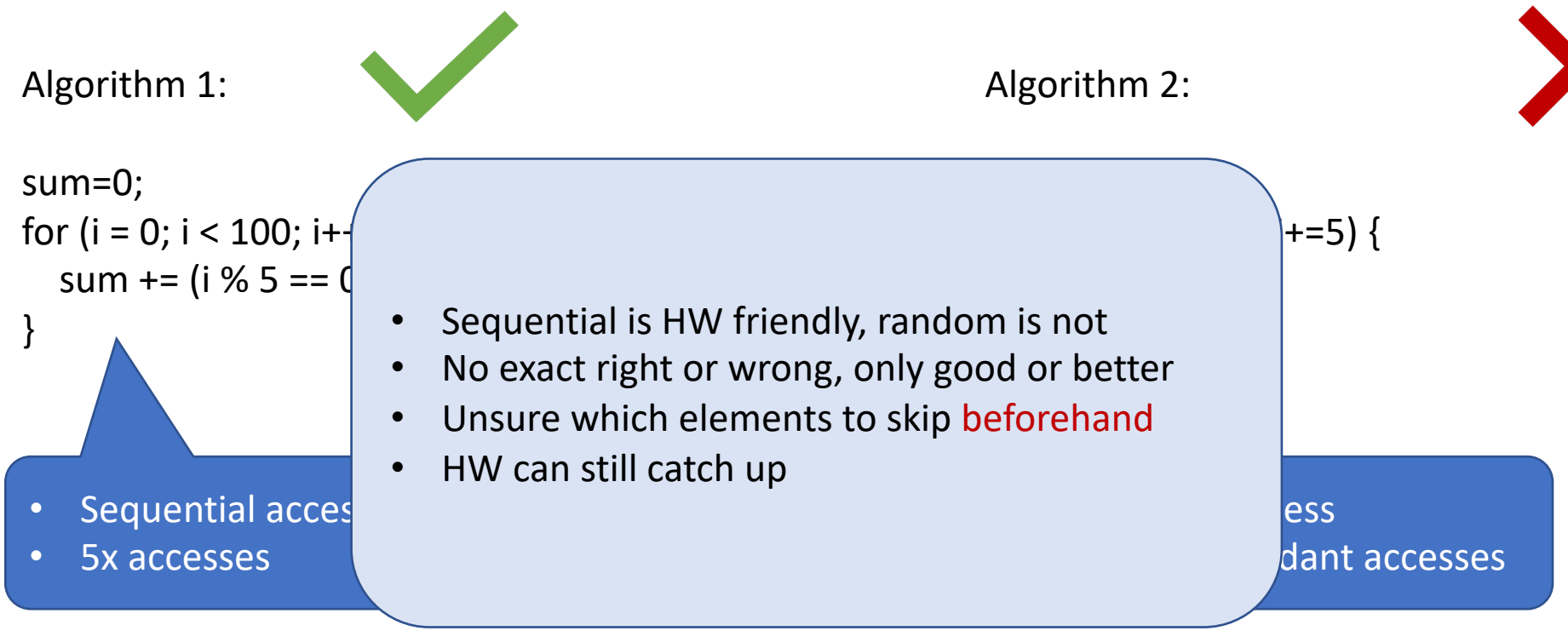
Algorithm 2:

```
sum=0;
for (i = 0; i < 100; i+=5) {
   sum += data[i];
}
```

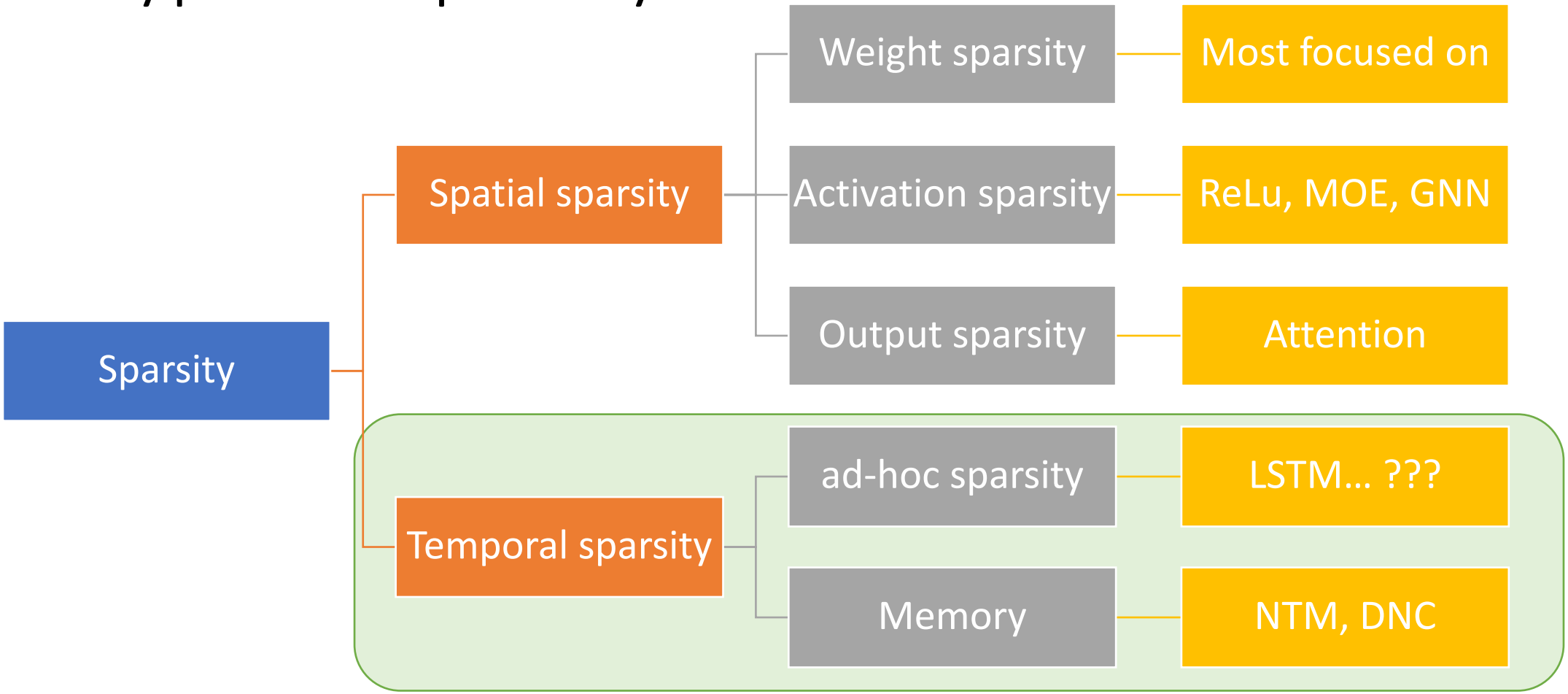
- Stride access
- No redundant accesses

The choice of algorithms, in deep learning

- Problem: get the sum of every 5th element in an 100-element array.



Types of sparsity



The blockers of sparsity

- Motivation: sparsity is to improve computation efficiency
  - -> prune from dense model
  - -> compare with quantization
- Methodology: sparse algorithm is immature
  - -> quality loss in the process of sparsification
  - -> prune from dense model
- Competition: hardware on dense compute is very efficient
  - -> hard bar to beat against
- Scalability: small models in the sweet spot of dense computation
  - -> the benefit of sparsity is not exposed

Follow the Trend

- Model size increase exponentially
  - GPT-3: 175B parameters[4]
  - Switch transformers: 1.6T parameters[5]
- Dense training cost explodes
  - GPT-3: $4.6M to train[6]
- Hardware cannot train long sequence transformers
  - Attention scales square to the sequence length.
  - Memory, compute are both challenge

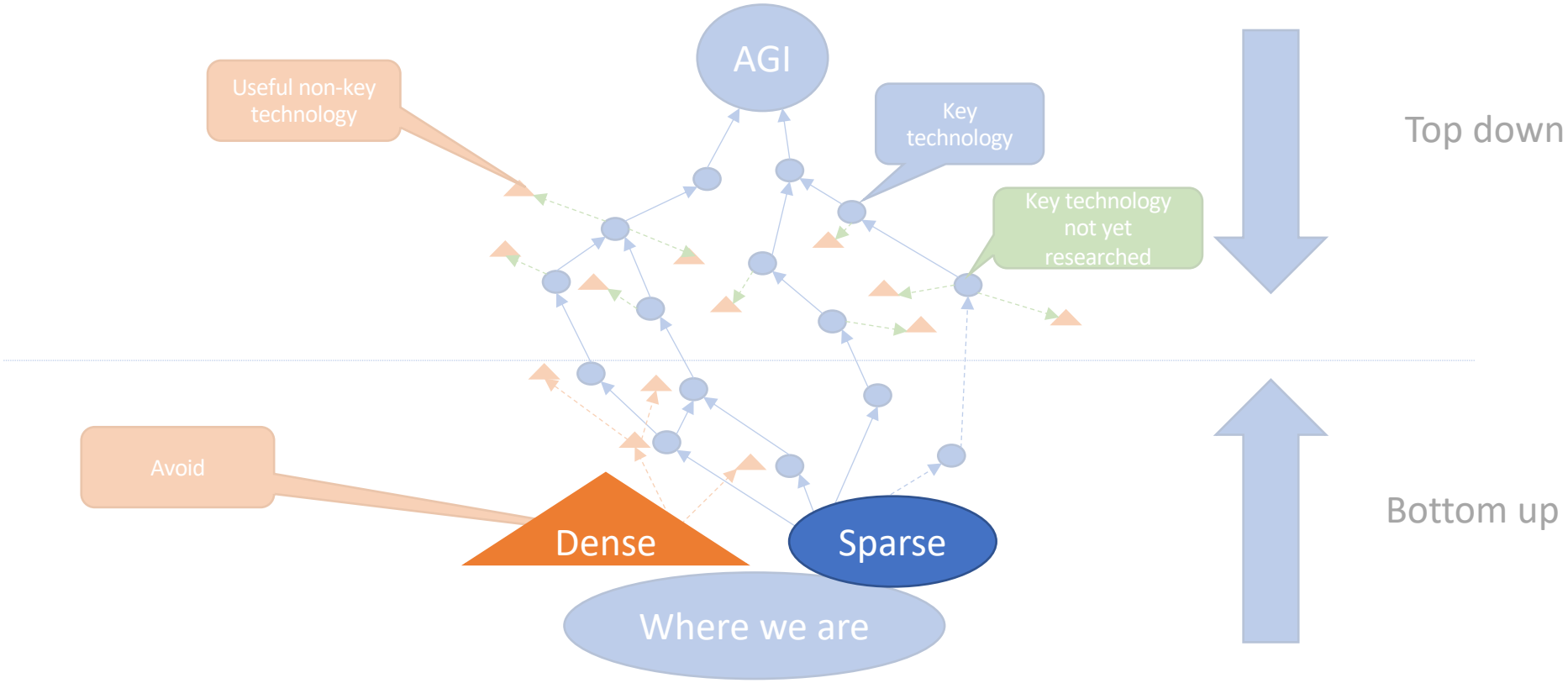
Follow the Trend

- Model size increase exponentially
  - GPT-3: 175B parameters[4]
  - Switch transformers: 1.6T parameters[5]
- Dense training cost explodes
  - GPT-3: $4.6M to train[6]
- Hardware cannot train long sequence transformers
  - Attention scales square to the sequence length.
  - Memory, compute are both challenge

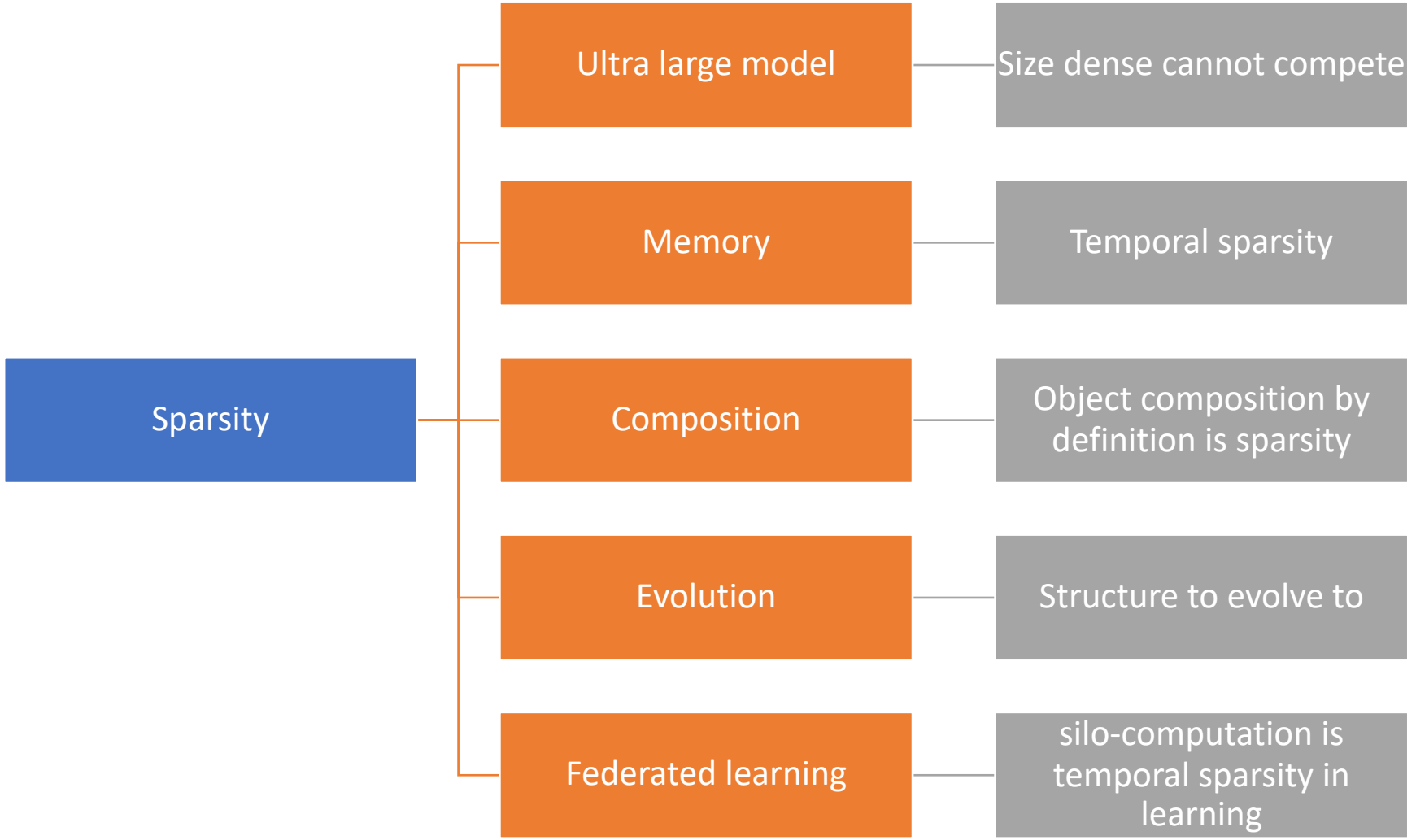
Dense hardware cannot keep up with the model advancement

- "Dennard scaling" is broken
  - Intel@ 2006 -> go multi-core
  - NVIDIA@ 2023?? -> go sparsity??

Path towards AGI



Importance of sparsity



Sparsity on Ultra Large Models

Sparsity on Ultra large models

- Problem statement
  - Explore sparsity to make possible the scale of models infeasible for dense computation
- Why do this first?
  - More people have realized the need (necessity)
  - Already has some research foundation (easiness)
  - Foundation for other researches (importance)
- Key blockers
  - Algorithms to find large and sparse models directly
  - Hardware to efficiently training large and sparse models
    - Require algorithms to show the benefit

Ultra large models: Algorithm

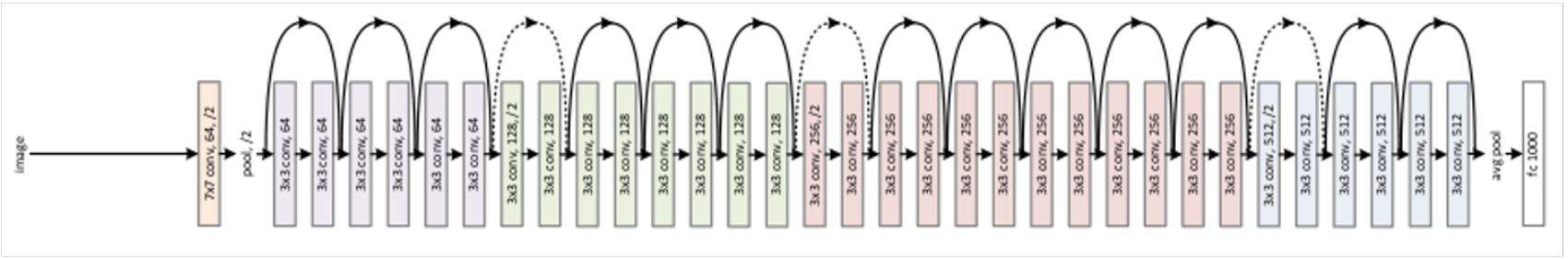
- Ultra large weight sparsity <- current focus
  - Most previous researches are focused on this
  - We have achieved an important milestone: GaP
  - Many follow up researches remain
- Ultra large activation/output sparsity <- next focus
  - Many people see the potential
    - Attention in transformers, mixture of experts
  - Few shot learning
  - Impact the hardware design more
    - Single batch training

Ultra large models: Hardware

- Need to support both weight sparsity and activation sparsity
  - Addressing only one of them is not complete (HW design skewed)
- Focus on single batch training
  - Activations in the same batch goes through different paths
  - Enable few shot learning
  - Existing approach increase parallelisms inside a batch, need to think the other way round
    - Attentions in transformers[9]
    - Mixture of experts[10]
    - Switch transformers[11]: compromise algorithm for efficiency

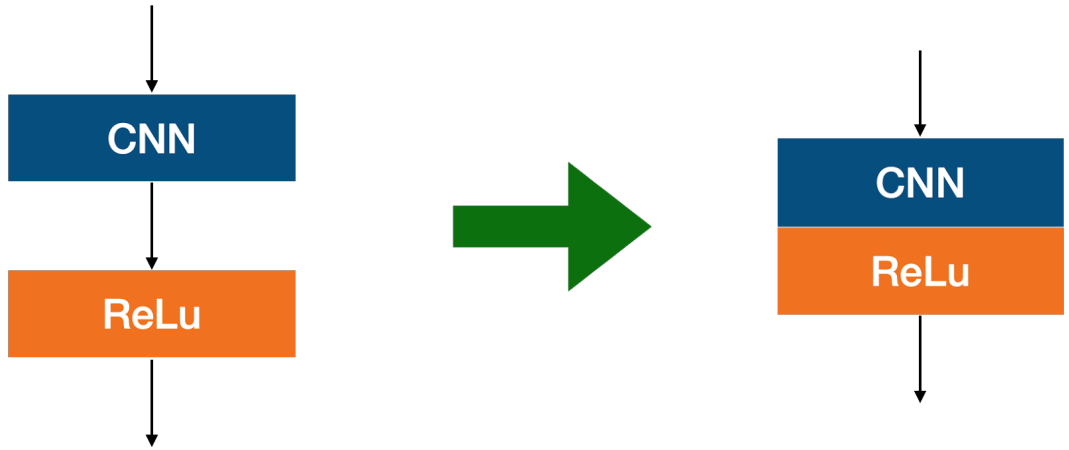
Data reuse in dense models

- Data reuse is the single most important factor in performance optimization
  - Improve data reuse in all aspects: algorithm/software/hardware
- Partition model execution layer after layer
- Within a layer, use tiling (algorithm) and cache (hardware) to facilitate data reuse



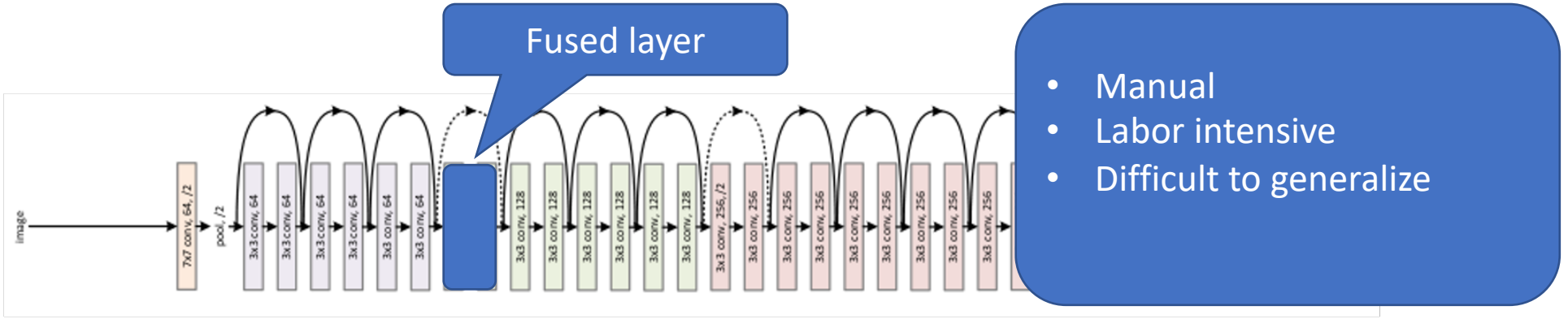
Data reuse in dense models

- Data reuse is the single most important factor in performance optimization
  - Improve data reuse in all aspects: algorithm/software/hardware
- Partition model execution layer after layer
- Within a layer, use tiling (algorithm) and cache (hardware) to facilitate data reuse
  - Fuse layers to improve cross-layer data reuse



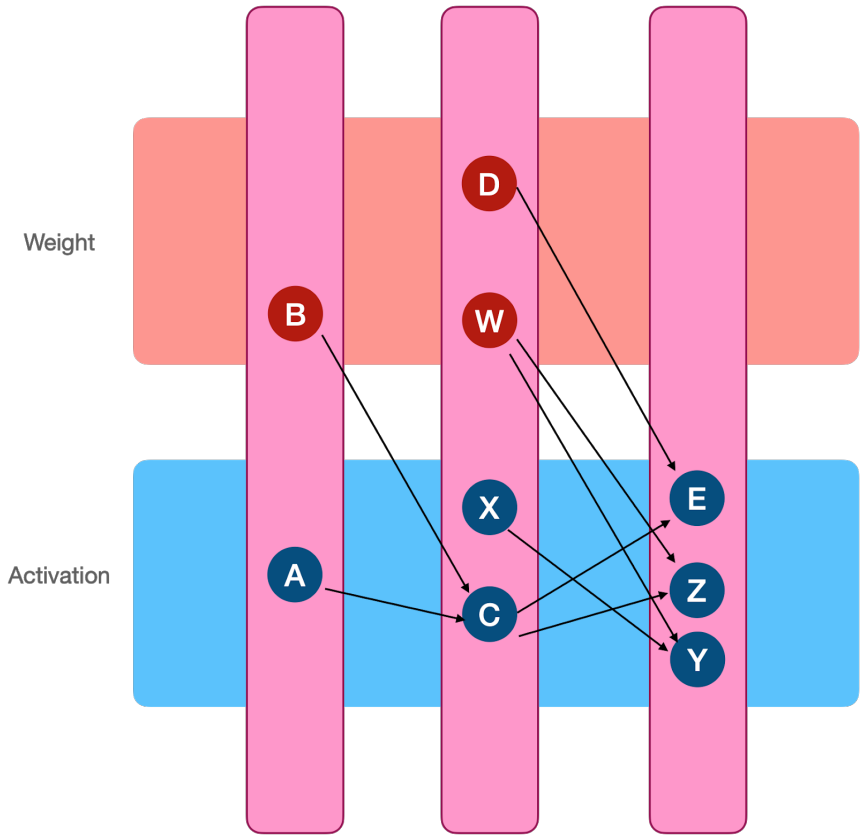
Limits on data reuse for dense models

- Layer-wise partition is NOT optimal for dense models
  - Cross layer data reuse is manually explored (fuse)
- Still reasonable for dense models, but disaster for sparse models
  - Dense: sequential data access within a layer, require large batch size
    - Leverage multithreading in CPU/GPU
  - Sparse: random data access within a layer/cross layers, worse for single batch



Data reuse in sparse models

- Data access is random, cannot explore spatial locality within a layer.
- Can explore input/output data locality
  - a + b = c; c + d = e; -> c can be reused



Data reuse in sparse models

- Data access is random, cannot explore spatial locality within a layer.
- Can explore input/output data locality
  - a + b = c; c + d = e; -> c can be reused

- Data flow is most important, control is not. Design is IO centric.
- Asynchronous programming (reactive programming, event-driven programming) !
  - Javascript, NodeJS

Semi-event driven programming

- Events, not from external users
  - Static: Execution path determined by weights and graph structure
  - Dynamic: Execution path determined by dynamic activation values
- Semi-event scheduling
  - Not responsive to external events (unknown to the scheduler)
  - Proactively schedules known events to maximize data reuse at runtime
    - The event completion time can be estimated (to some extent)
  - Non-blocking IO/memory access
- Good for single batch training

Software impact of event driven programming

- Event is a programming model exposed to the programmers to improve efficiency and coding productivity
  - Programmers do not need to coordinate load/store and execution
  - Not using multithreading (directly). Not sequential. Difficult for software programmers to understand
- Can be encapsulated within a framework
  - Models can still be represented as sparse models
- Framework design is more difficult
  - Fuzzy boundary between framework (graph) and kernel
  - Compiler is the way to go

Existing hardware architecture support for events

- In-order processors
  - Strictly sequential processing
- Out-of-order processors: limited scope event handling capability
  - FSM: scoreboarding, non-blocking cache
  - Event: tomosulo
- Stream processing: data centric
  - Good for static graph. How to handle dynamic graph?
- Graph accelerator: most closely related

Existing hardware architecture support for events

- In-order processors
  - Strict
- Out-of-
  - FSM
  - Even
- Stream
  - Good
- Graph a

Current GPU/CPU are not sufficient

Need to co-design fine-grained event driven programming models and architectures

Hardware exposes non-blocking accesses through ISA so that software can perform global scheduling

Asynchronous circuit

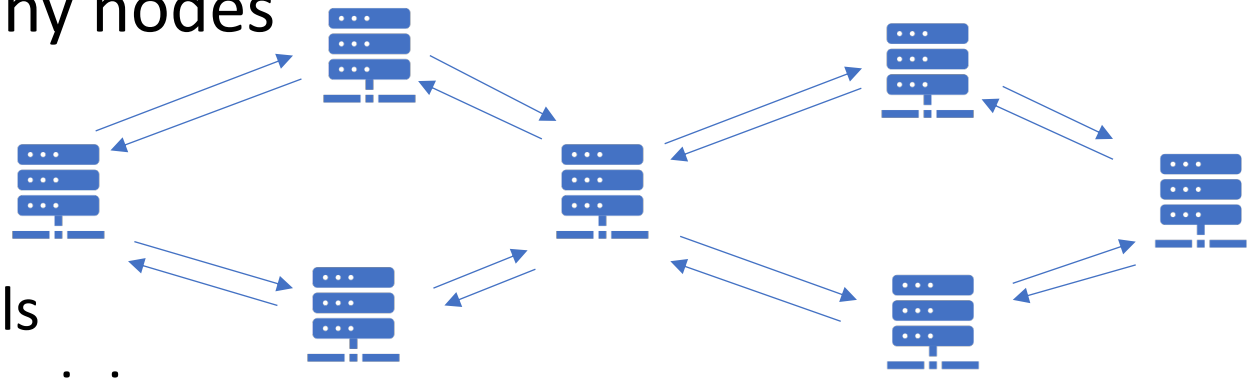
- If the entire stack is asynchronous, do we need synchronous sequential circuit?

- Asynchronous circuit
  - Pros: high performance, low power, purely asynchronous
  - Cons: difficult to design, not scalable in current EDA system

- Very long term goal

Distributed event driven programming

- Large models are distributed to many nodes
- Each node handles events on itself
- Require asynchronous SGD
  - May be required on ultra-large models
  - May not be an issue for single batch training



- Is this one model training or many models collaboratively learning??
  - The boundary is blurred
  - Another step towards ultra-large models

Current stack

- Sequential concept -> sequential algorithm -> sequential software programming model-> sequential hardware architecture -> sequential circuit

Future sparse stack

- Sequential concept -> sequential algorithm -> sequential software programming model-> sequential hardware architecture -> sequential circuit

- Asynchronous concept -> asynchronous algorithm -> sequential software programming model-> sequential hardware architecture -> sequential circuit

Future sparse stack

- Sequential concept -> sequential algorithm -> sequential software programming model-> sequential hardware architecture -> sequential circuit

- Asynchronous concept -> asynchronous algorithm -> asynchronous software programming model-> sequential hardware architecture -> sequential circuit

Future sparse stack

- Sequential concept -> sequential algorithm -> sequential software programming model-> sequential hardware architecture -> sequential circuit

- Asynchronous concept -> asynchronous algorithm -> asynchronous software programming model-> asynchronous hardware architecture -> sequential circuit

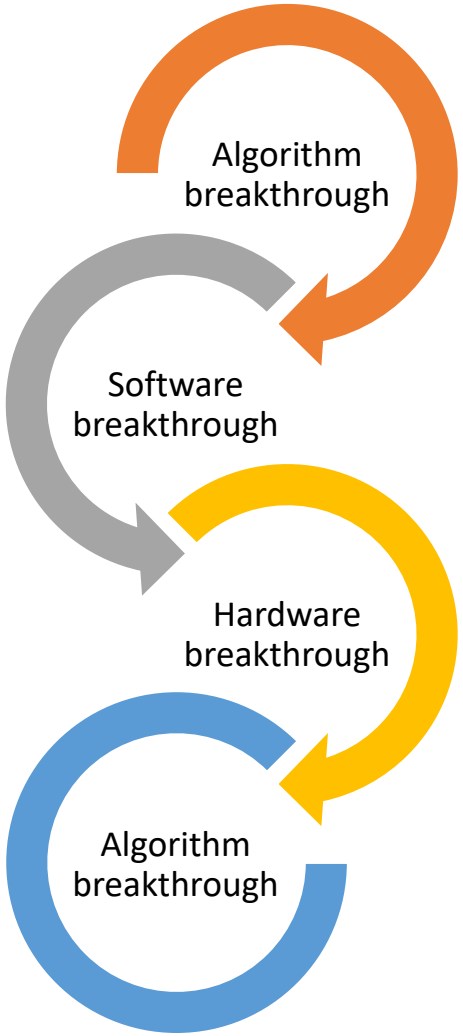
Future sparse stack

- Sequential concept -> sequential algorithm -> sequential software programming model-> sequential hardware architecture -> sequential circuit

- Asynchronous concept -> asynchronous algorithm -> asynchronous software programming model-> asynchronous hardware architecture -> asynchronous circuit

Algorithm-software-hardware co-design

First algorithm show the feasibility of sparsity, but performs poorly in existing hardware



Then hardware is enhanced to efficiently process the new algorithm, enabling research not possible before

Then software is enhanced to efficiently process the new algorithm, enabling research not possible before

Based on the new hardware design, new algorithm can be found to explore more advanced algorithms

Sparsity on memory, composition, and evolution

Sparsity on memory

- Problem statement
  - Selectively retrieve relevant past features to assist the prediction of current feature.

- Memory is temporal sparsity
  - Save and retrieve relevant features in the past
    - Even though the past features may be saved continuously, the retrieval is random
    - Can be viewed as some kind of attention mechanism
  - May spatial sparsity provide hints on the memory mechanism?

Sparsity on composition

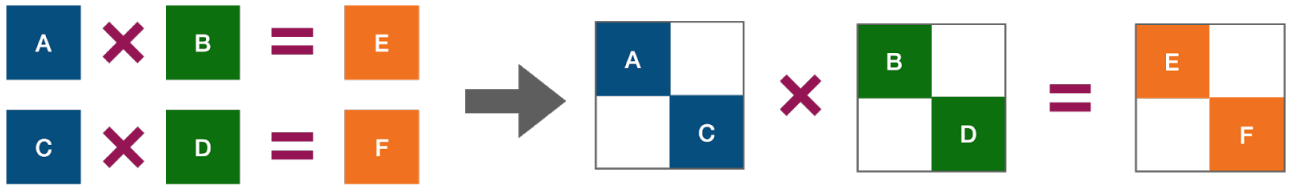
- Can we find one methodology that sparsifies all of the following?
  - Weights within a layer
  - Weights across layers
  - Activations within a layer
    - Attention, MOE
  - Activations across layers
  - Historical data points (memory)

- We cannot do it on dense computation (search structure), but can we do it in sparse computation?

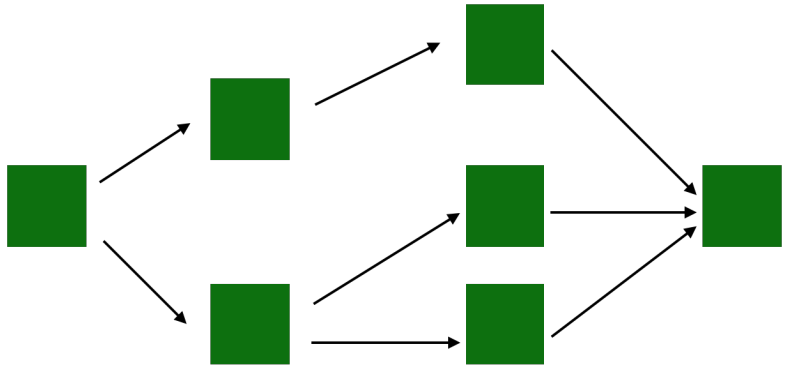
- One composition strategy
  - One way to decide “relevance”

Sparsity on evolution

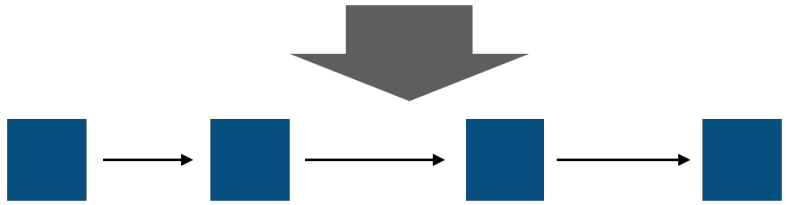
- Sparsity is a structure



NAS on dense models



NAS on sparse models



Sparsity on Evolution

- Evolution needs to start from a good structure
- Hypothesis: evolution on sparsity is easier than dense
- Hypothesis: evolution embedding for sparsity is more representative than dense

Few shot learning

- Not few shot learning in fine-tune stage of transfer learning
  - New data still much less than transfer learning (< 100)
- Require single batch training
- Can only be successful with memory
  - Every new data point needs to be compared with the previously learned data points
  - Currently previous learned data points are saved in the model via pretraining
- Can only be fully successful with composition
  - Need to de-compose new data to existing known data

AGI, Starting From Myself

Outline

- Top down view of the path breakdown towards AGI
  - The ability to learn
  - The ability to evolve
  - The ability to scale
  - Composition
- Sparsity is an essential step towards AGI
  - Why sparsity is important
  - Sparsity on ultra large models
  - Sparsity on memory, composition, and evolution
- AGI, starting from myself
  - People's view on AGI
  - AGI in academia and industry
  - Interesting near term projects

People’s view on AGI

- AGI doesn’t exist – Yann LeCun[20]
- Bottom up approach
  - AI is a moving target
  - “As soon as it works, no one calls it AI anymore” – John McCarthy[2]
- Our approach: Top down and bottom up approach
  - Relatively fixed target
  - Identify key technical barriers
  - Use great leap forward approach

AGI in academia and industry

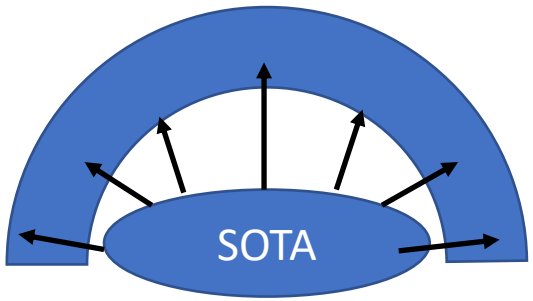
- Academia
  - Forward looking, but lack of funding
  - DARPA: AI next campaign
    - $2B on contextual reasoning
- Industry
  - Hardware companies: Intel, AMD, NVIDIA
    - AI follower, do not lead algorithm innovation
  - Application companies: Facebook
    - Innovative in AI algorithm, but lack hardware support
    - Cannot complete multiple rounds of advancements

AGI in academia and industry

- Full stack companies
  - Google
    - ML Algorithm: transformers, mixture of experts, depthwise convolution
    - Software: MLIR, XLA
    - Hardware: TPU v1/2/3
  - DeepMind
    - #1 AI research institution: AlphaGo, AlphaStar, AlphaFold …
    - Focus on reinforcement learning
      - Not key technology IMO
- Alibaba
  - Full stack company
  - Long term research ambition: DAMO academy
  - Possible to be #1 in AI research

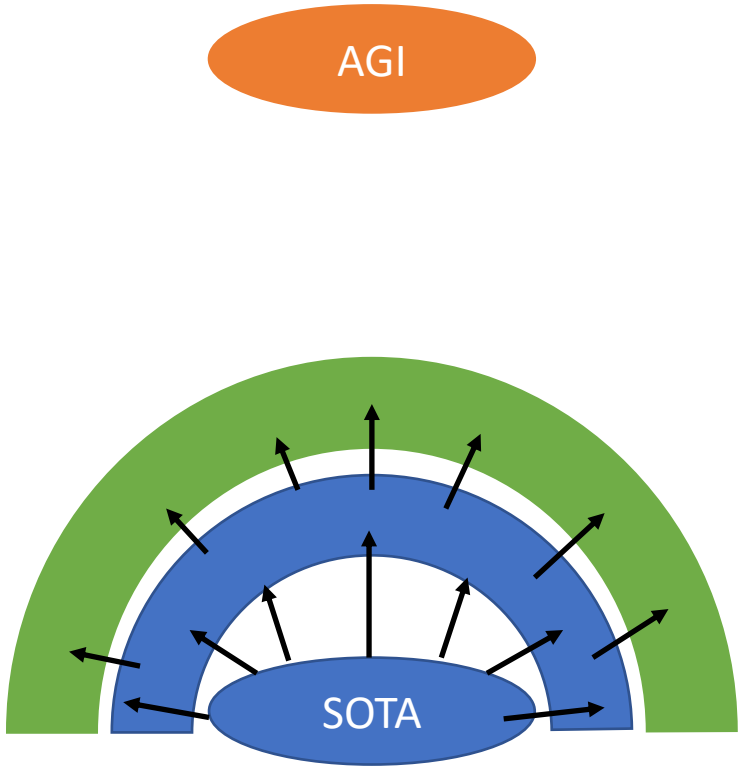
Research towards AGI

Research towards AGI



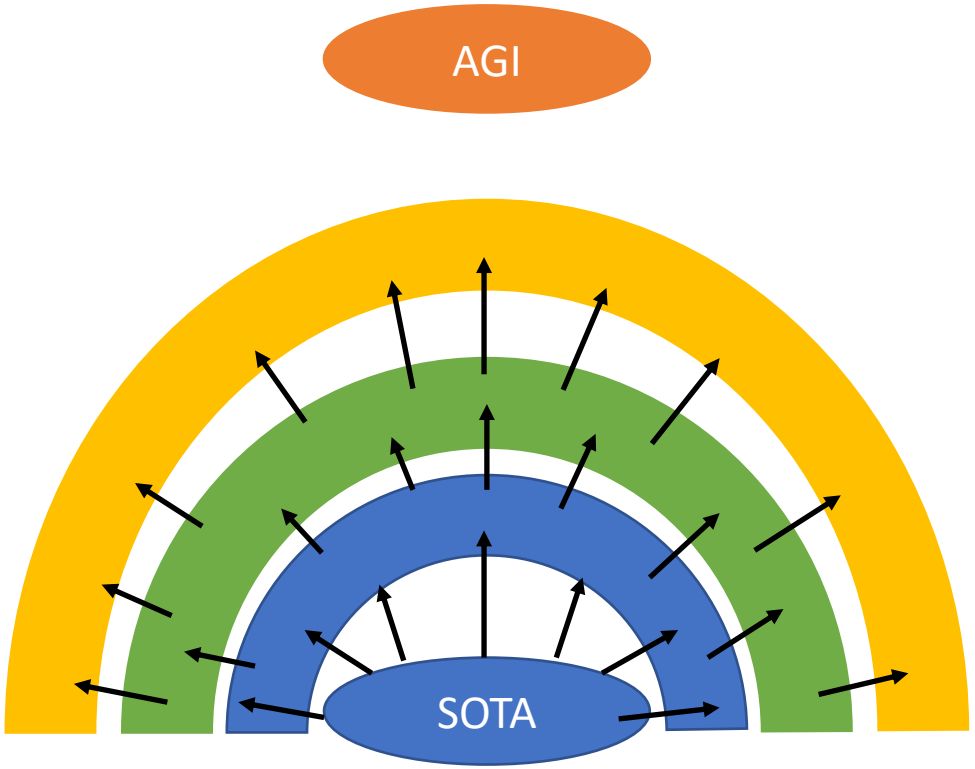
Bottom up approach

Research towards AGI



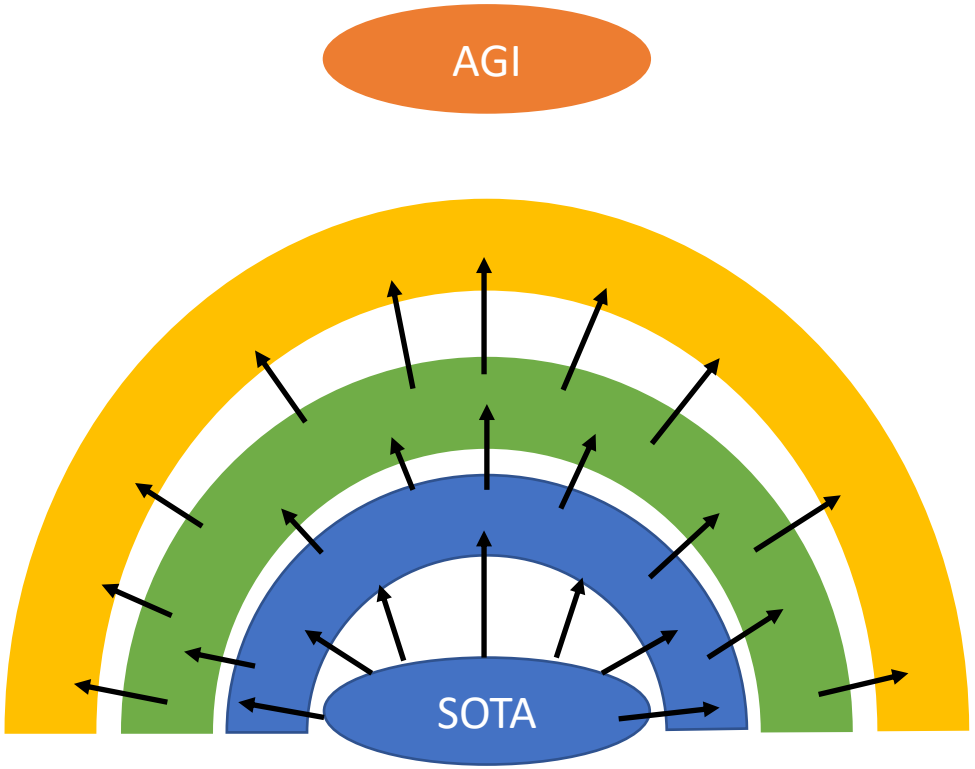
Bottom up approach

Research towards AGI

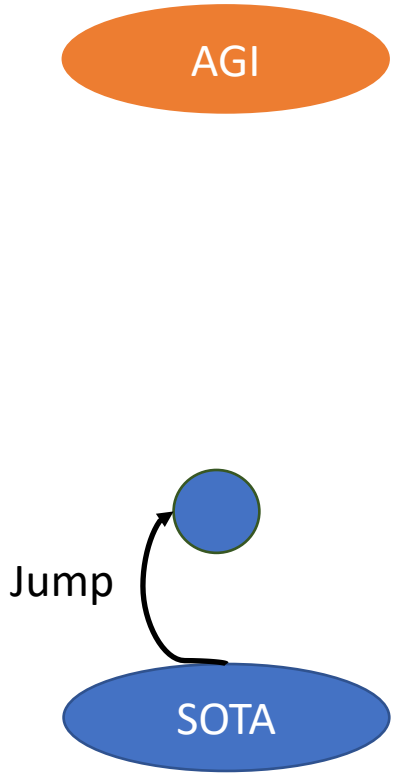


Bottom up approach

Research towards AGI

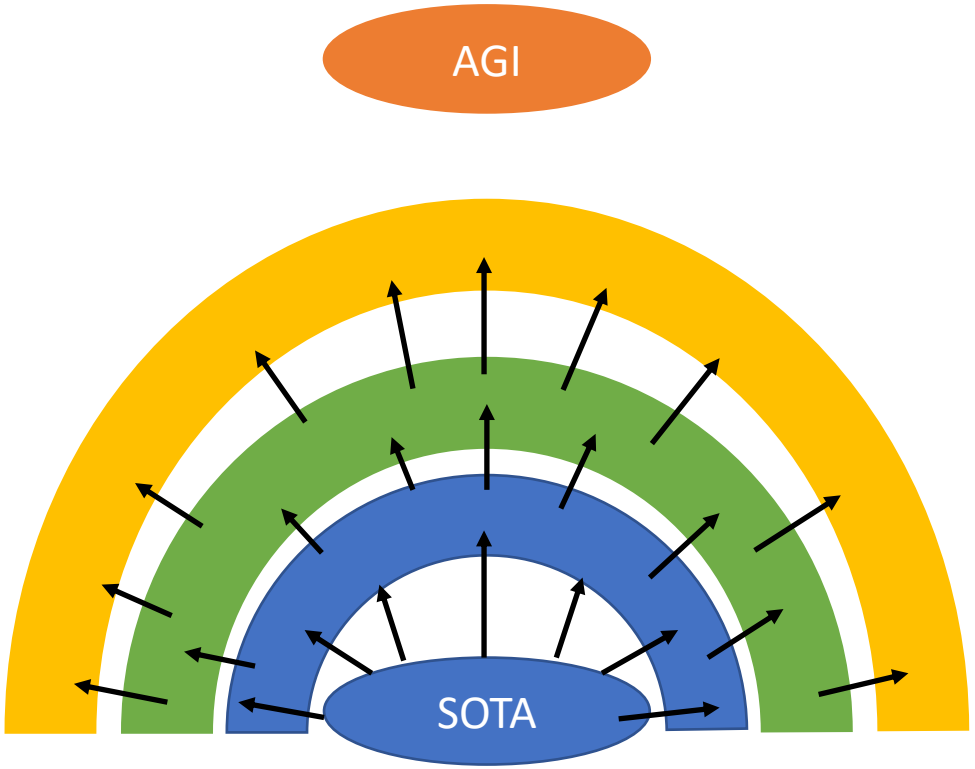


Bottom up approach

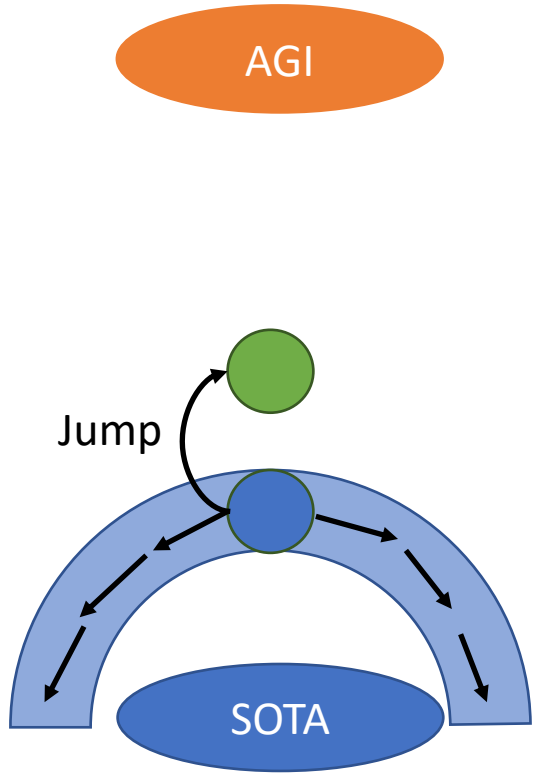


Key technology approach

Research towards AGI

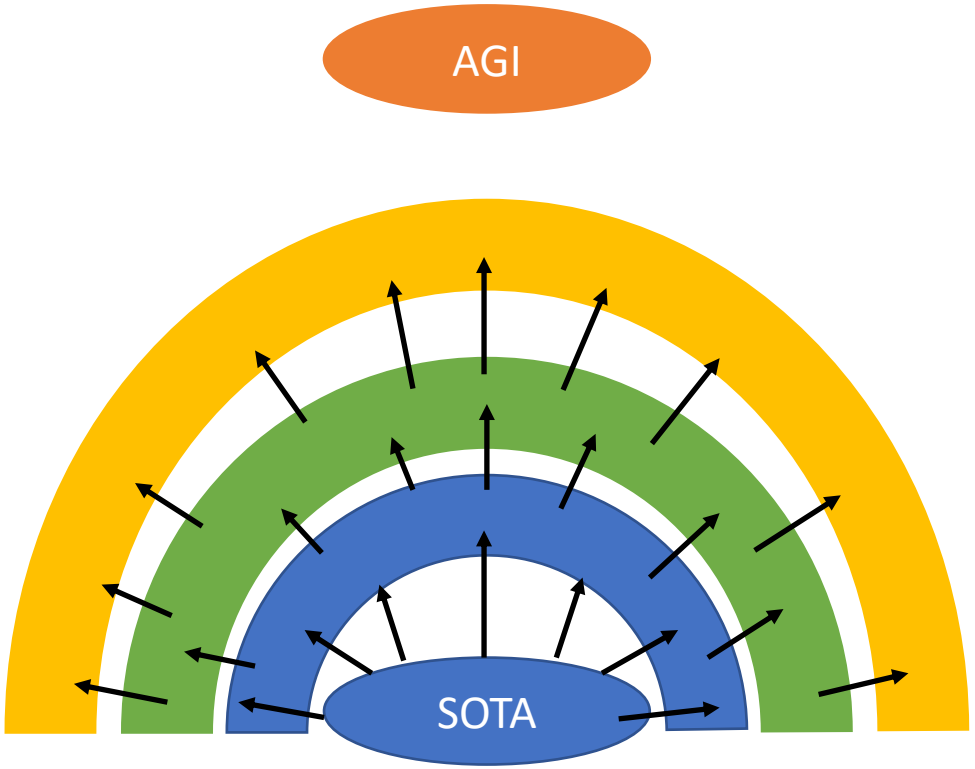


Bottom up approach

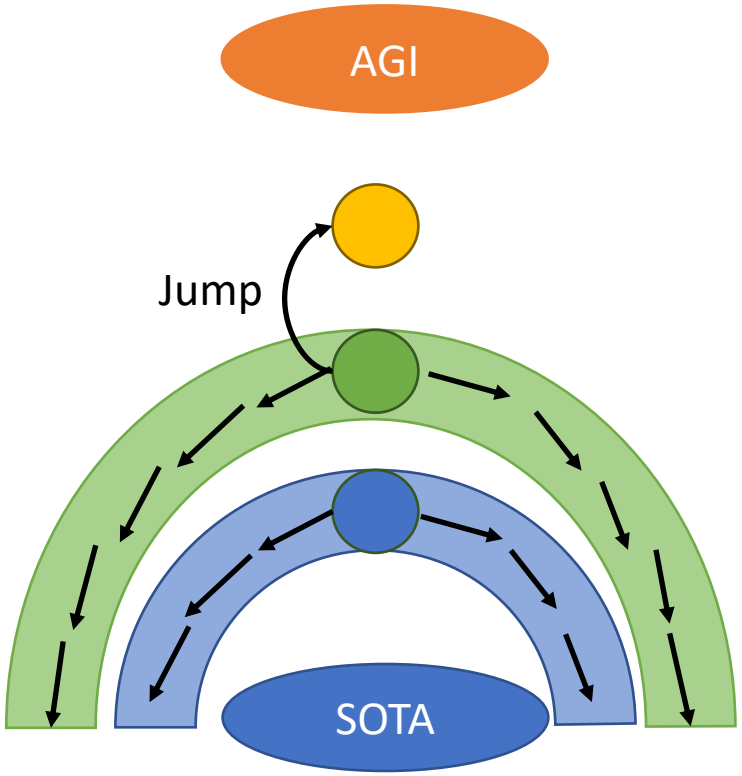


Key technology approach

Research towards AGI

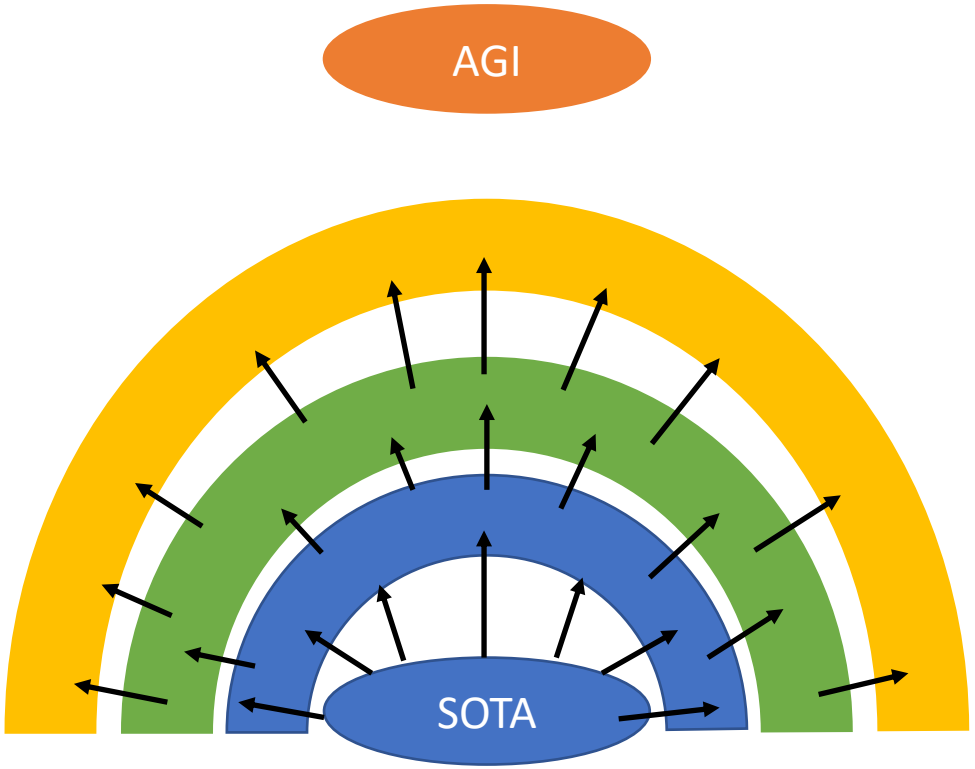


Bottom up approach

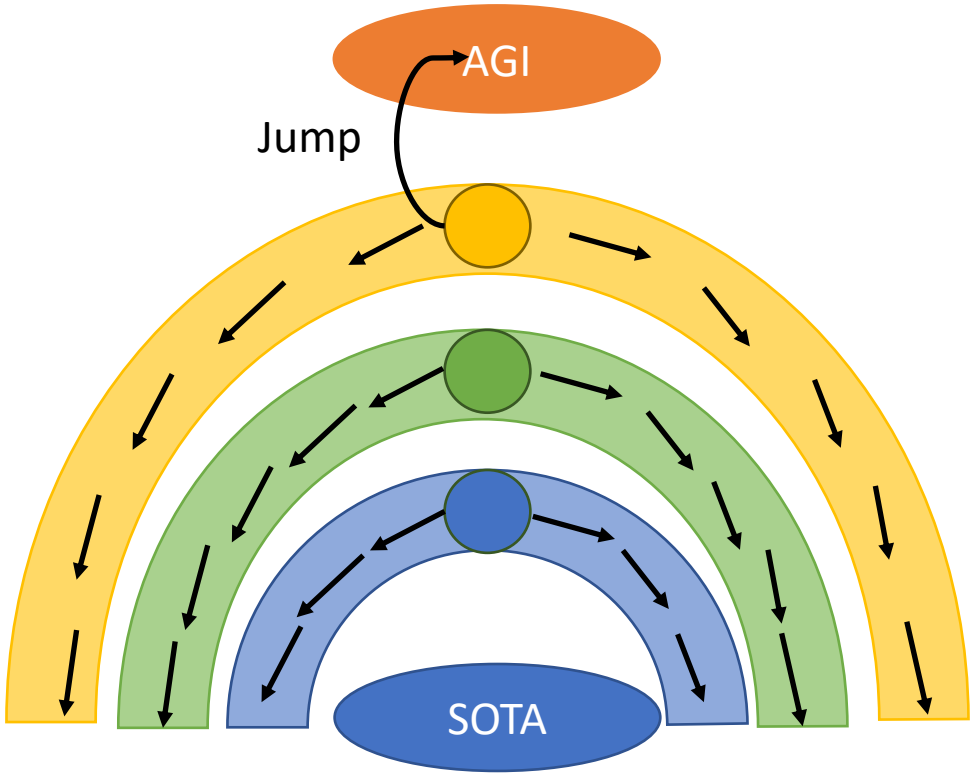


Key technology approach

Research towards AGI

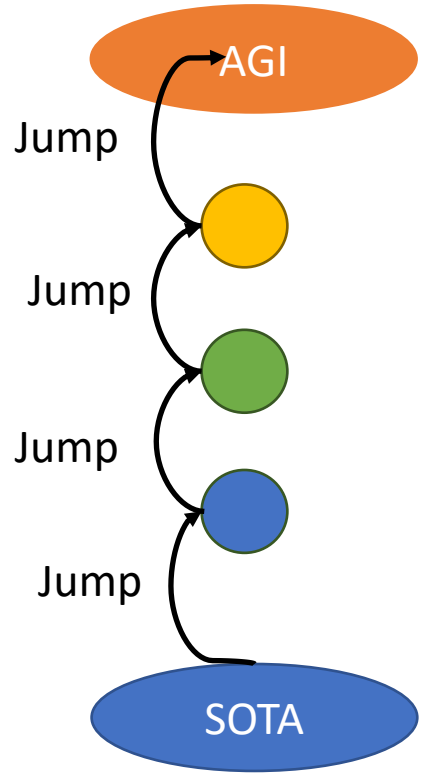


Bottom up approach



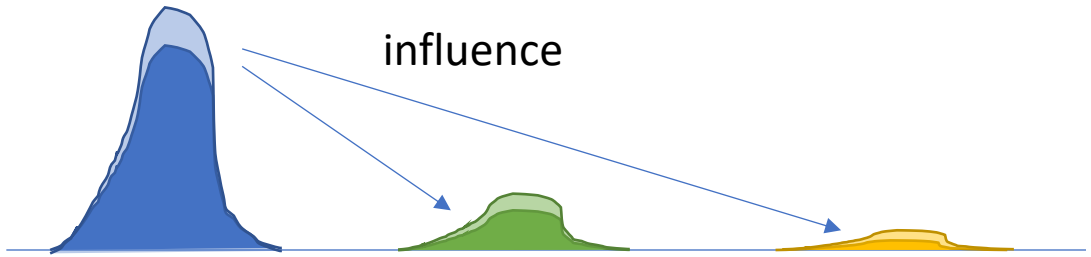
Key technology approach

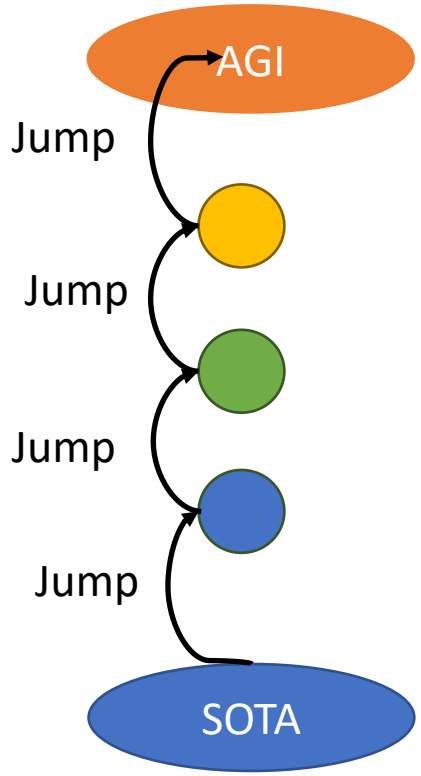
What we research



Key technology approach

Collaboratively research technical milestones





Key technology approach

Possible Near Term Projects

Key Technical Milestone: Ultra Large Sparse Model

Sparsity Algorithm Research

- Train a large and sparse model without dense model
  - Fine grained, coarse grained
  - Weights, activation

Asynchronous Hardware-Software Co-design

- Long term research
  - Programming model
  - Software framework
  - Hardware architecture/microarchitecture design

Sparsity on Transformers

Transformer

- Transformer is important
  - Widely used in NLP, penetrating to CV, recommendation, etc.
  - The secret sauce is the self attention
- Transformer is large
  - GPT-3, switch transformers
  - Likely the first model exceeding the compute limit
- Transformer is compute intensive
  - Attention complexity scales n^2 with long sequence attention
    - Not scalable for long sequences

Heated research area

- Self-attention is low rank
  - Very few elements in the attention matrices are relevant

- Very hot area to reduce self-attention complexity
  - Many sparsify the attention matrices
    - Different from weight sparsity, the relevant locations are input dependent
  - Big Bird[13], Longformer[14], Deformable DETR[15], and many more

Explore algorithm/software/hardware additions to efficiently executing sparse attention for transformers

Sparsity on Mixture of Experts

Mixture of Experts

- Mixture of experts (MOE) is coarse grained activation sparsity
  - Select the “experts” based on inputs

- MOE is the only way to train ultra-large models
  - Google: switch transformers[11] ~1.6T parameters
  - Alibaba: M6, ~100B parameters

- MOE imposes system challenges
  - Data parallelism, pipeline parallelism, model parallelism

Main cause of system challenges

- Large batch size contributes to the system challenges
  - Inertia from training small models
  - Large batch size exposes more data parallelism on existing identical hardware with uniform processing
- MOE is non-uniform processing on different hardware
- Should we impose large batch size?

Data pipeline single batch training

- Use asynchronous programming model
- Model training is performed in an asynchronous way
- Each server behaves reactively
- May need asynchronous SGD
  - Need to ensure convergence rate
- Software change may boost performance quite a bit
  - Encapsulate within framework

- Hardware enhancement will boost yet another level
  - Need to figure out what software is not capable

Key Technical Milestone: Memory

Memory is important

- Can reduce dependence on large amount of data
  - Reduce data labeling cost
  - Reduce training cost
- Current status
  - Embed memory in model
  - Check “all” past events
- Not much research on location based memory
  - NTM, DTC

Extended Research: Sparsity on Federated Learning

Federated Learning 1.0

- Problem statement
  - **Train** a deep learning model **weights** across multiple **decentralized** devices holding **local** data samples without exchanging them.
- Challenges
  - **Communication/computation cost of edge devices**
  - **Non-iid**
  - Security
  - Data privacy

Federated Learning 2.0

- Problem statement
  - **Evolve** a deep learning model **structure** across multiple **decentralized** devices holding **local** data samples without exchanging them
    - With centralized coordination
    - Without centralized coordination
- Challenges
  - Challenges for federated learning 1.0
  - **Heterogeneous model structures on non-iid data**
  - **Infrequent decentralized information exchange**

Why Sparsity

- Communication/computation cost of edge devices
  - Sparse computation is more efficient
  - Sparse data communication is more efficient
- Non-iid
  - Sparsely connected super models
- Heterogeneous model structure on non-iid data
  - Sparsity is a model structure
    - Model structure exploration -> sparse model training
- Infrequent decentralized information exchange
  - Sparse patterns do not require frequent exchanges

Current Research/Industry Status

- Still predominantly on federated learning 1.0
  - Success stories
    - Google: Gboard
    - Apple: Siri
  - Emerging research directions
    - Autonomous driving
    - Smart city

- **No one is working on federated learning 2.0**
  - Our opportunities

Theory Research

Possible theory research

- Asynchronous SGD
- Beyond linear approximation
- Composition

Conclusion

- Top down view of the path breakdown towards AGI
  - The ability to learn, the ability to evolve, the ability to scale
  - Composition
- Sparsity is an essential step towards AGI
  - Why sparsity is important
  - Sparsity on ultra large models: use asynchronous programming
  - Sparsity on memory, composition, and evolution
- AGI, starting from myself
  - Alibaba is positioned to make breakthroughs in AI
  - Some projects worth starting now

梦想还是要有的，万一实现了呢？

一群有情有义的人，在一起做一件有价值意义的事

此时此刻，非我莫属

Will you join the venture?

References

- [1] https://en.wikipedia.org/wiki/Perception
- [2] https://en.wikipedia.org/wiki/Factory_method_pattern
- [3] https://hazyresearch.stanford.edu/software2
- [4] https://arxiv.org/abs/2005.14165
- [5] https://arxiv.org/abs/2101.03961
- [6] https://bdtechtalks.com/2020/08/17/openai-gpt-3-commercial-ai/
- [7] https://arxiv.org/abs/1410.5401
- [8] https://deepmind.com/blog/article/differentiable-neural-computers
- [9] https://arxiv.org/pdf/2012.09852.pdf
- [10] https://arxiv.org/pdf/1701.06538.pdf
- [11] https://arxiv.org/pdf/2101.03961.pdf
- [12] https://openai.com/blog/ai-and-compute/

References

- [13]: https://arxiv.org/pdf/2007.14062.pdf
- [14]: https://arxiv.org/pdf/2004.05150.pdf
- [15]: https://openreview.net/pdf?id=gZ9hCDWe6ke
- [16]: https://deepmind.com/about
- [17]: https://www.darpa.mil/work-with-us/ai-next-campaign
- [18]: https://feisun.org/2017/12/24/a-few-predictions-on-artificial-intelligence/
- [19]: https://feisun.org/2017/12/24/some-scribble-of-things/
- [20]: https://twitter.com/ylecun/status/1204038764122632193
- [21]: https://cacm.acm.org/magazines/2012/1/144824-artificial-intelligence-past-and-future/fulltext