GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

COMPUTATIONAL CELL ANALYTICS

Vision Transformers & Vision Foundation Models

Segment Anything & Applications in Microscopy

Constantin Pape

Institut für Informatik, Georg August Universität Göttingen

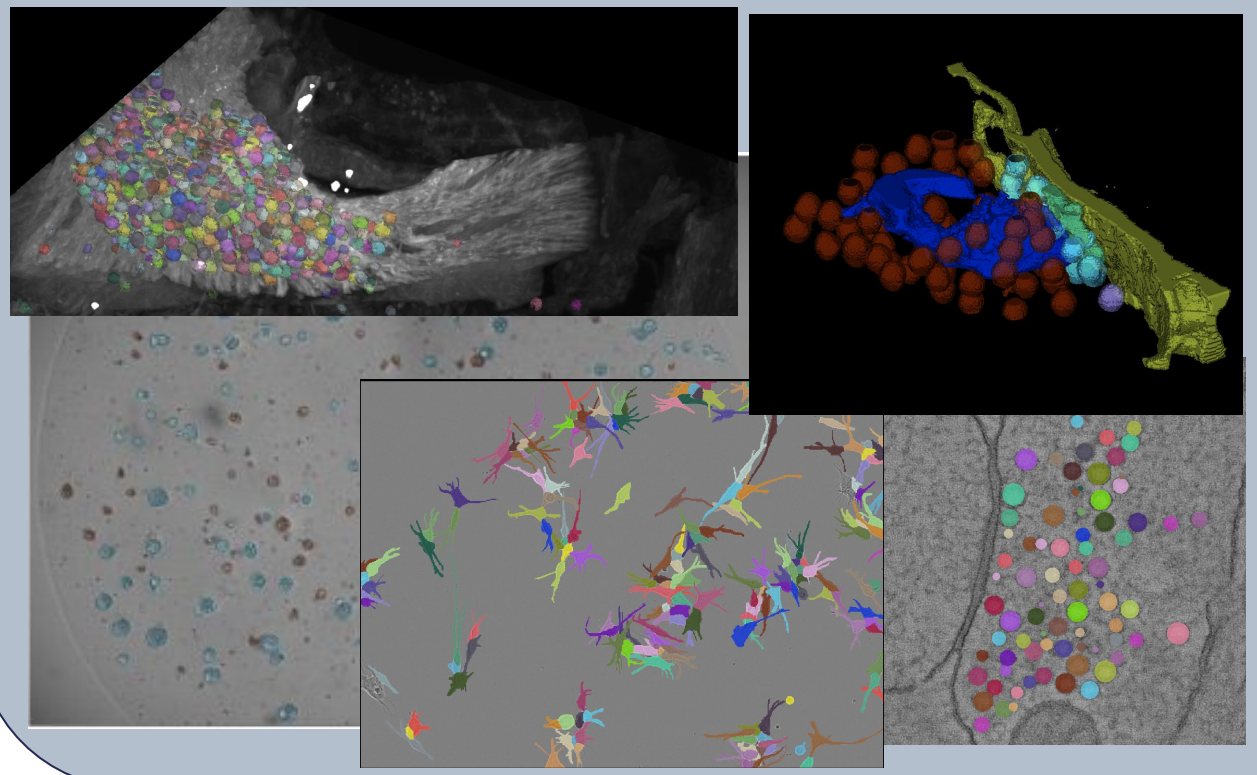
ccpape

https://user.informatik.uni-goettingen.de/~pape41/

Group vision: from images to insight and clinical relevance in collaboration with life scientists

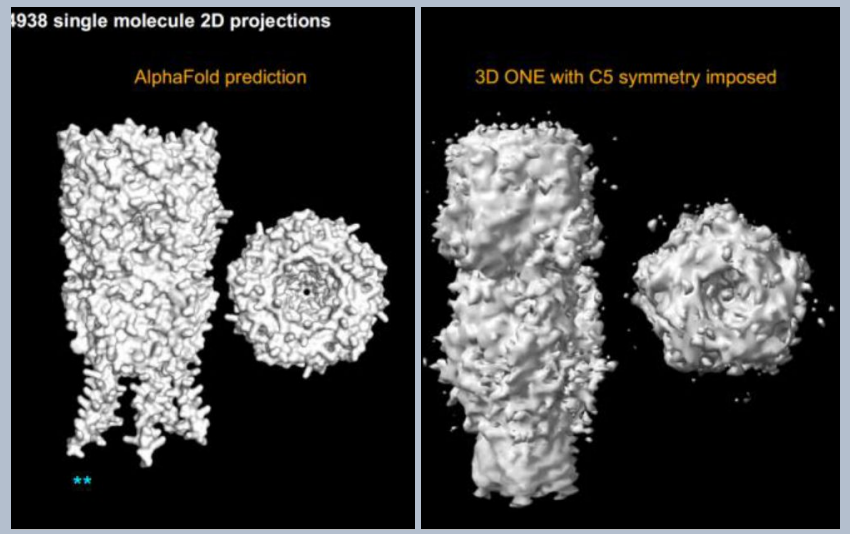
What we do …

Segmentation and tracking lots of stuff in microscopy



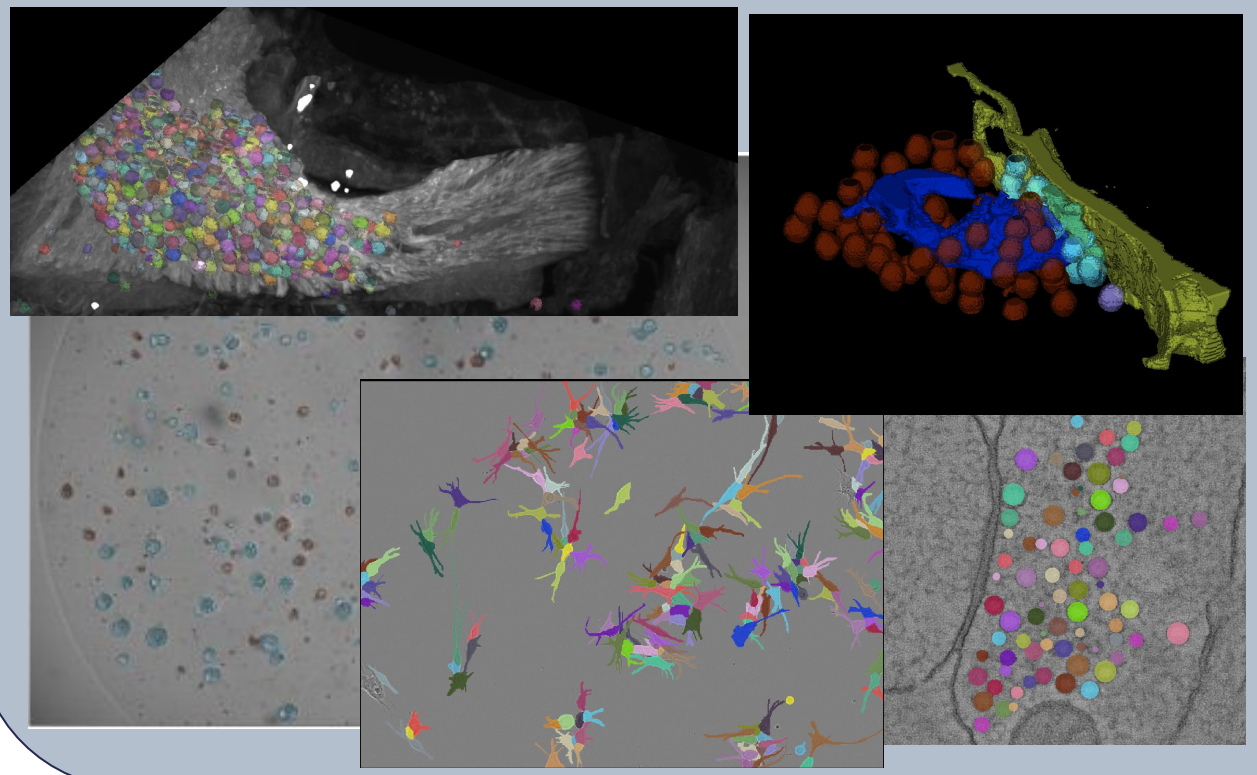
Representation learning for microscopy and multi-modal data

Protein structure analysis in cryo ET and **optical microscopy**



Group vision: from images to insight and clinical relevance in collaboration with life scientists

Segmentation and tracking lots of stuff in microscopy



Feasible with established DL methods
(U-Net -> CellPose / StarDist, …)
IF sufficient annotations / ground-truth!

Data annotation is a big bottleneck!

“Zero-shot” generalization is limited
-> need retraining unless their training data is quite similar

Vision Transformers & Vision Foundation Models

Large language models

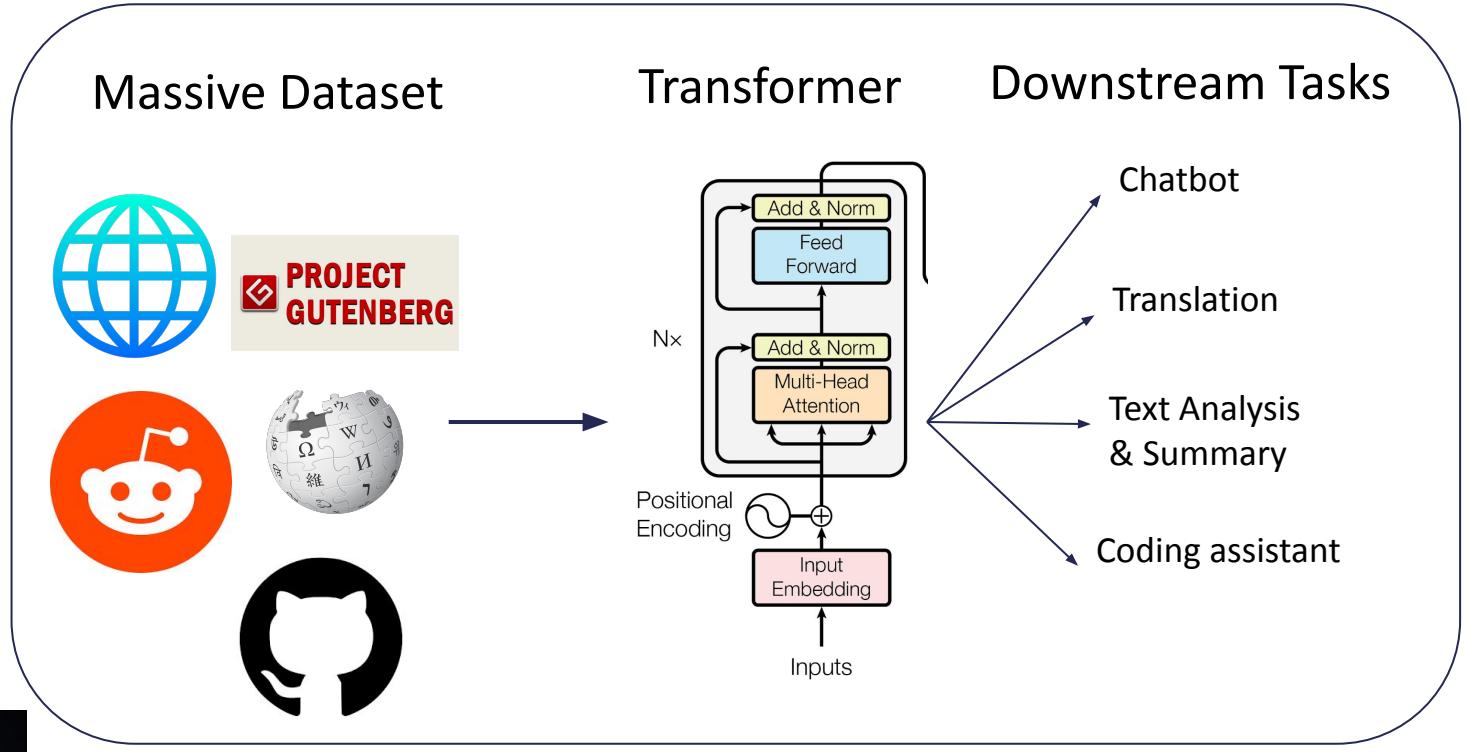
Large language models

ChatGPT

ANTHROPIC CLAUDE 3

LLama 3

Gemini

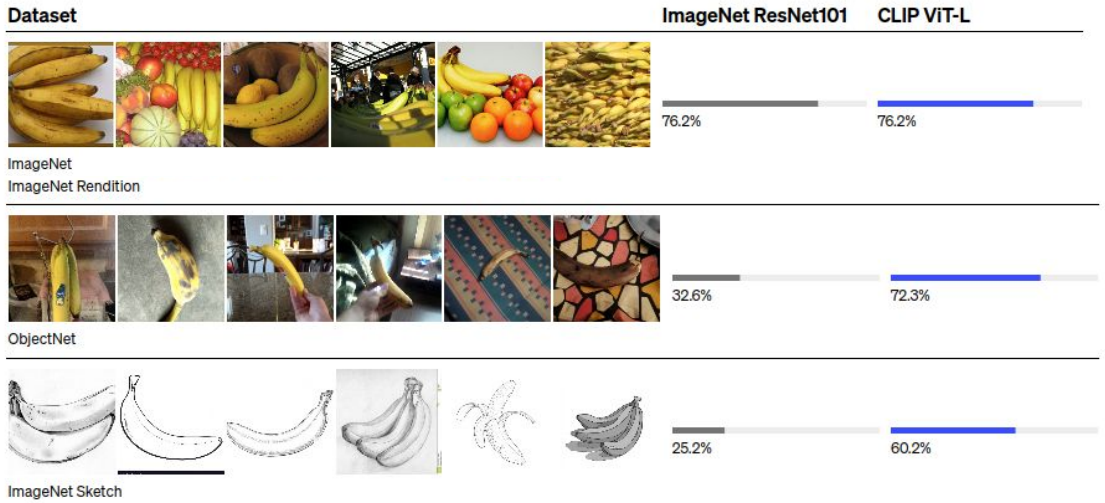


Can we do the same for vision?

Vision foundation models

Segment Anything

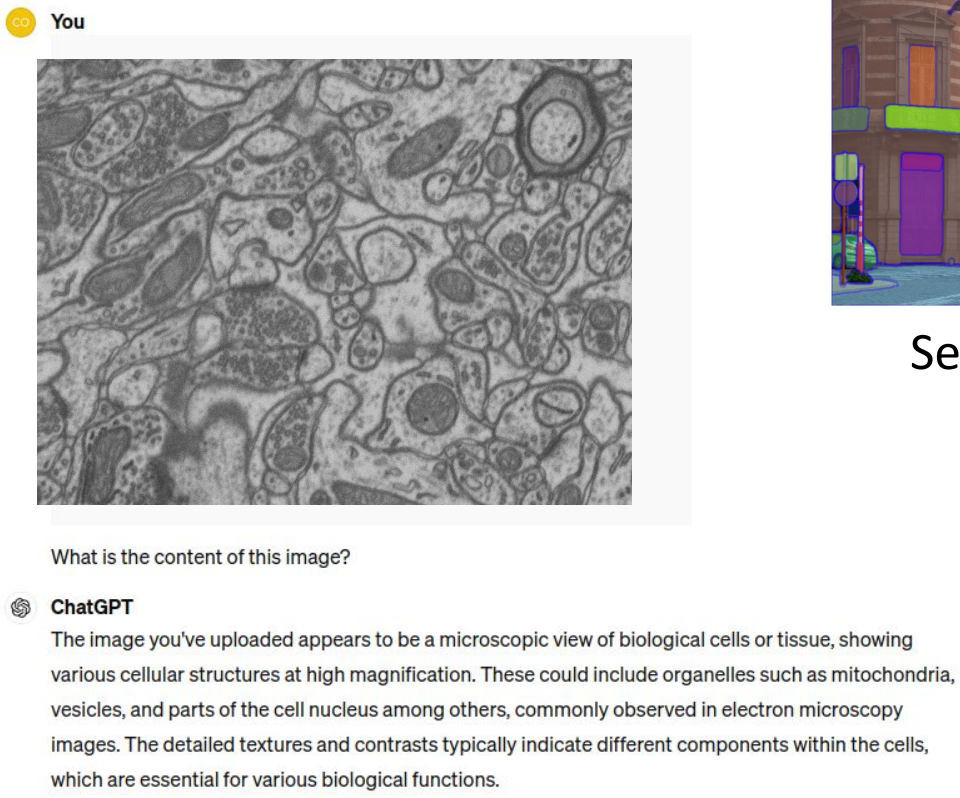
CLIP: Connecting text and images



Can we do the same for vision?

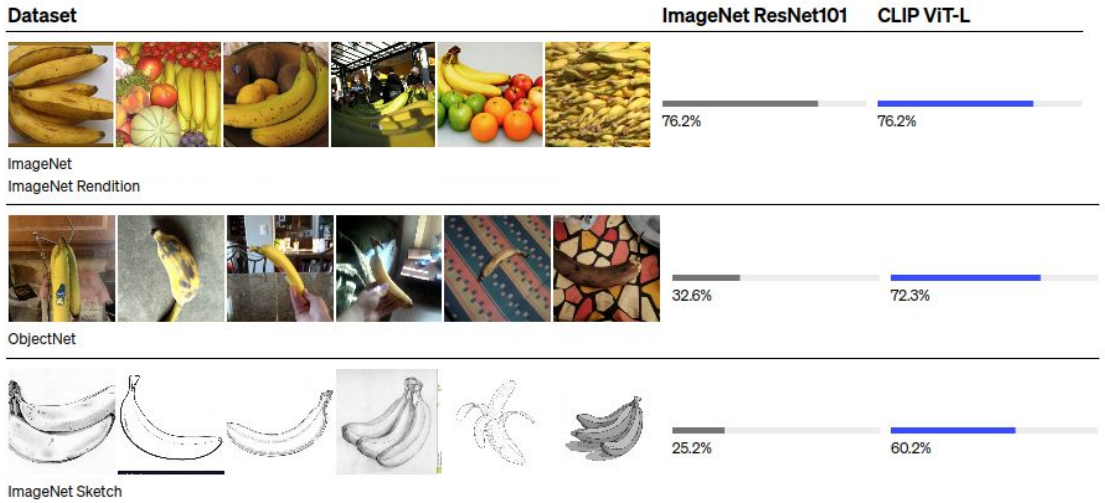
Vision foundation models

GPT4 Vision



Segment Anything

CLIP: Connecting text and images

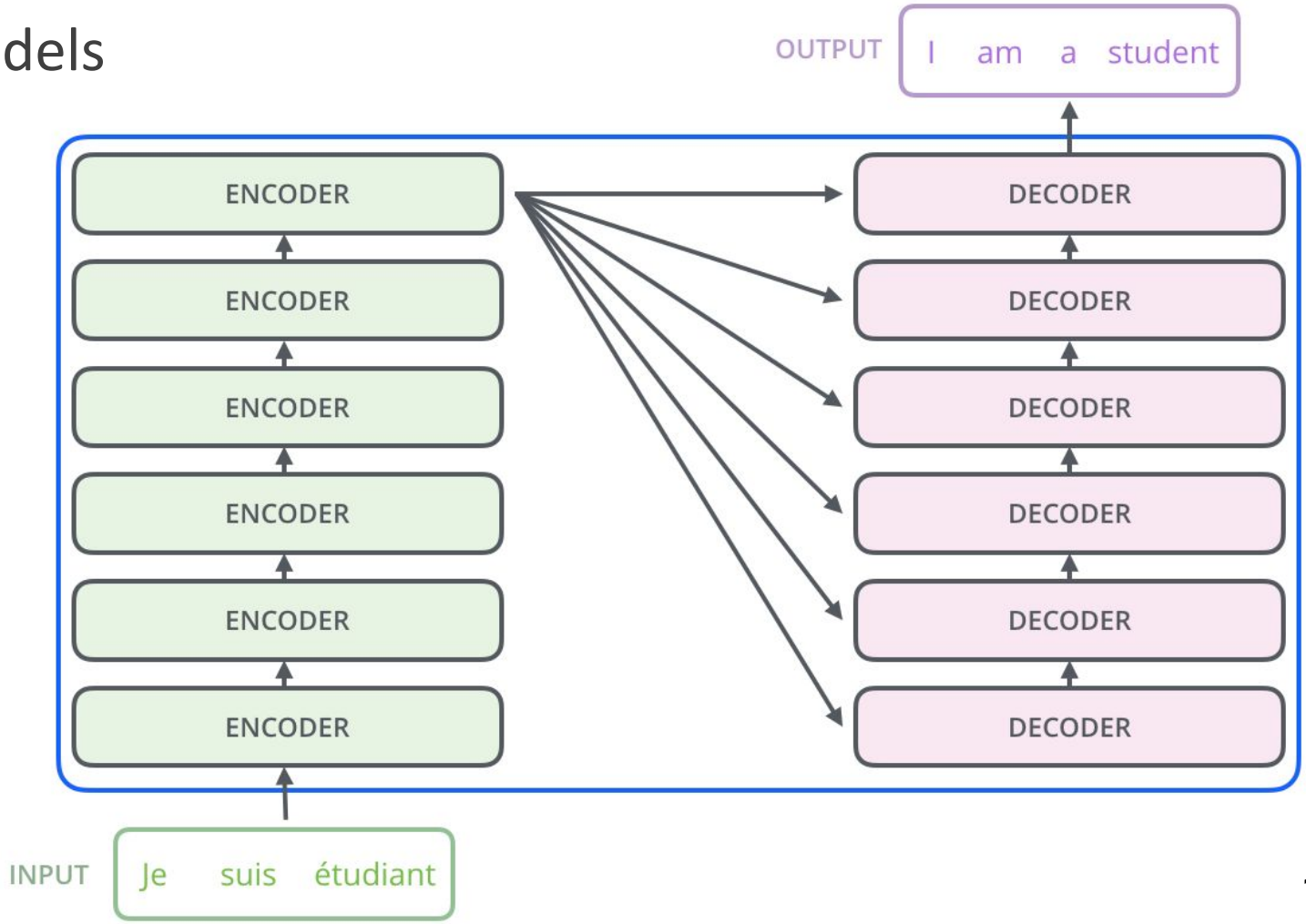


What is a transformer?

Transformer

Sequence to sequence models

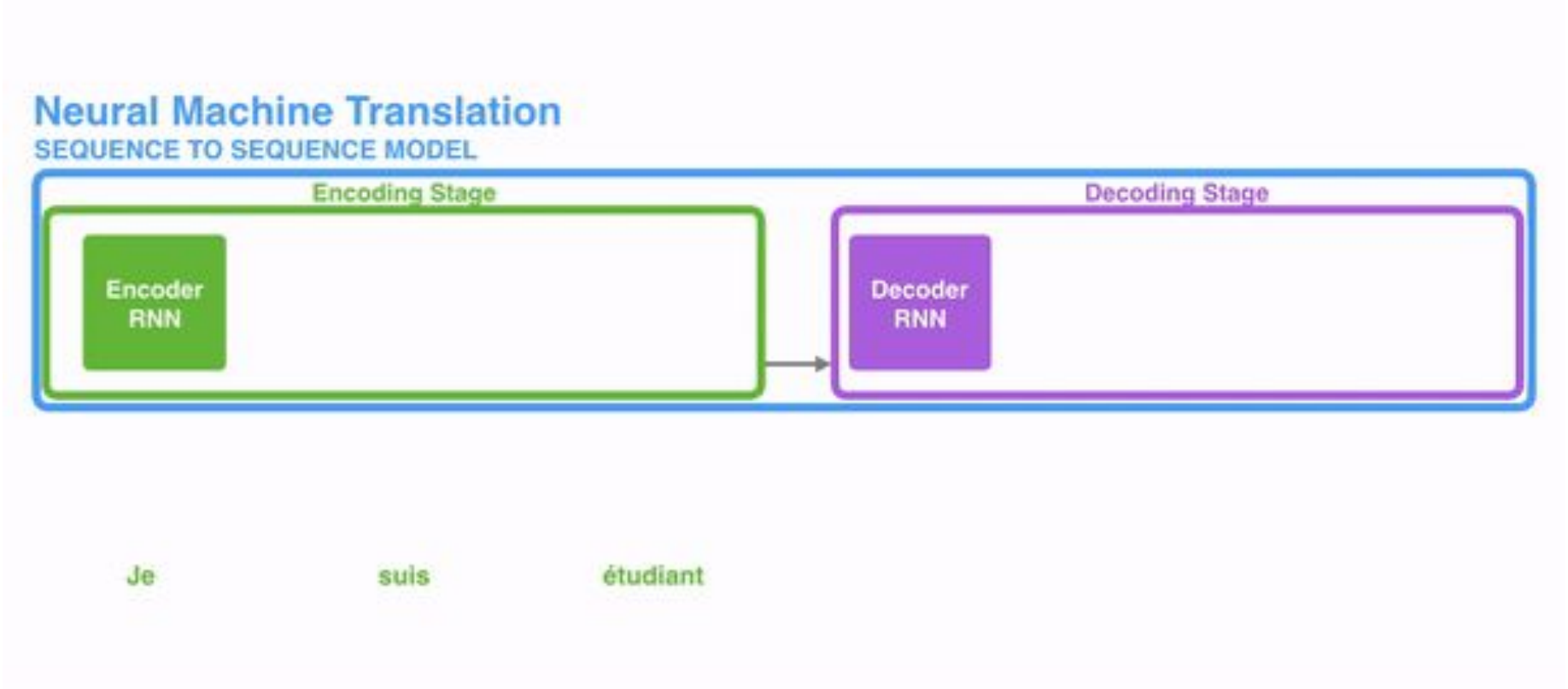
- Translation
- Text generation
- …



Why transformers?

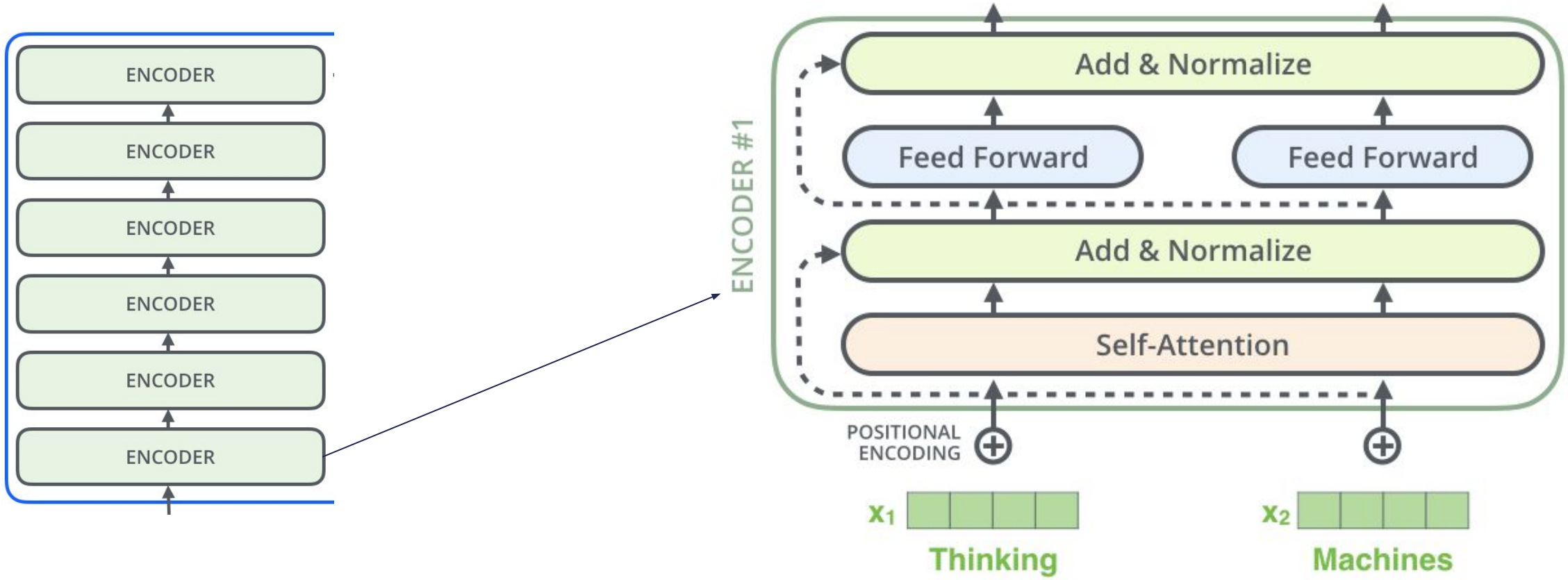
Previously: Recurrent Neural Networks (RNNs) like LSTM

- Problem: information decays over steps / “not enough memory”



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Transformer Block

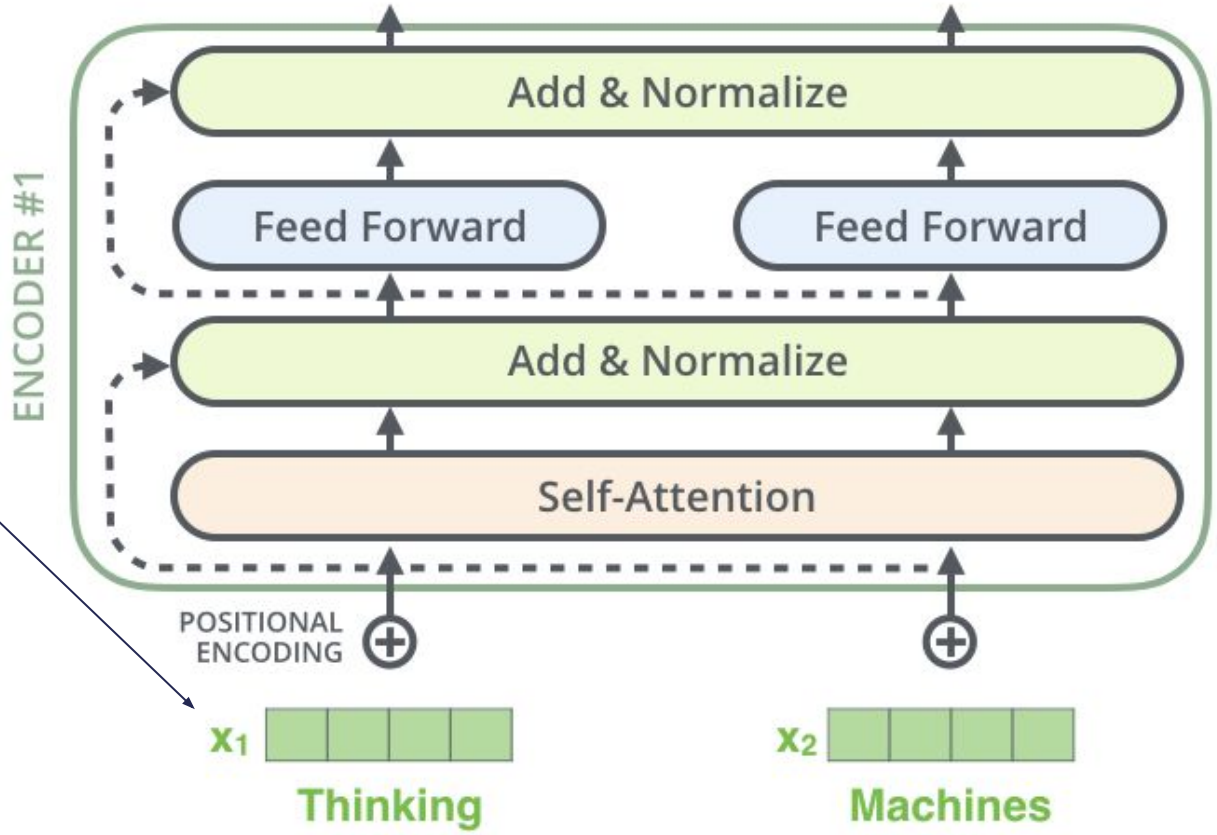


https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Transformer Block

The input: “tokenize” elements of your input sequence: compute a vector for each element

Problem dependent!

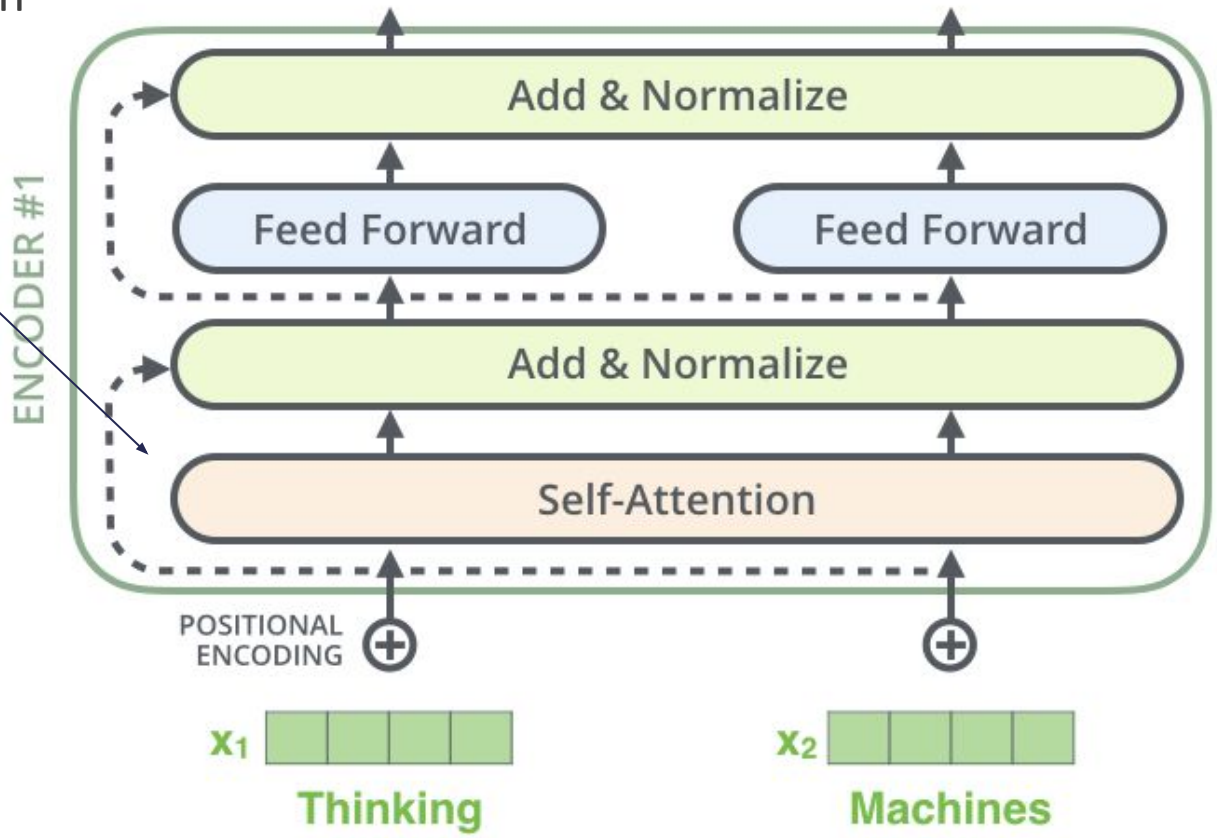


https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Transformer Block

Self-attention: propagate information between tokens, all-to-all connectivity

Attention: learns “affinities” between pairs of elements

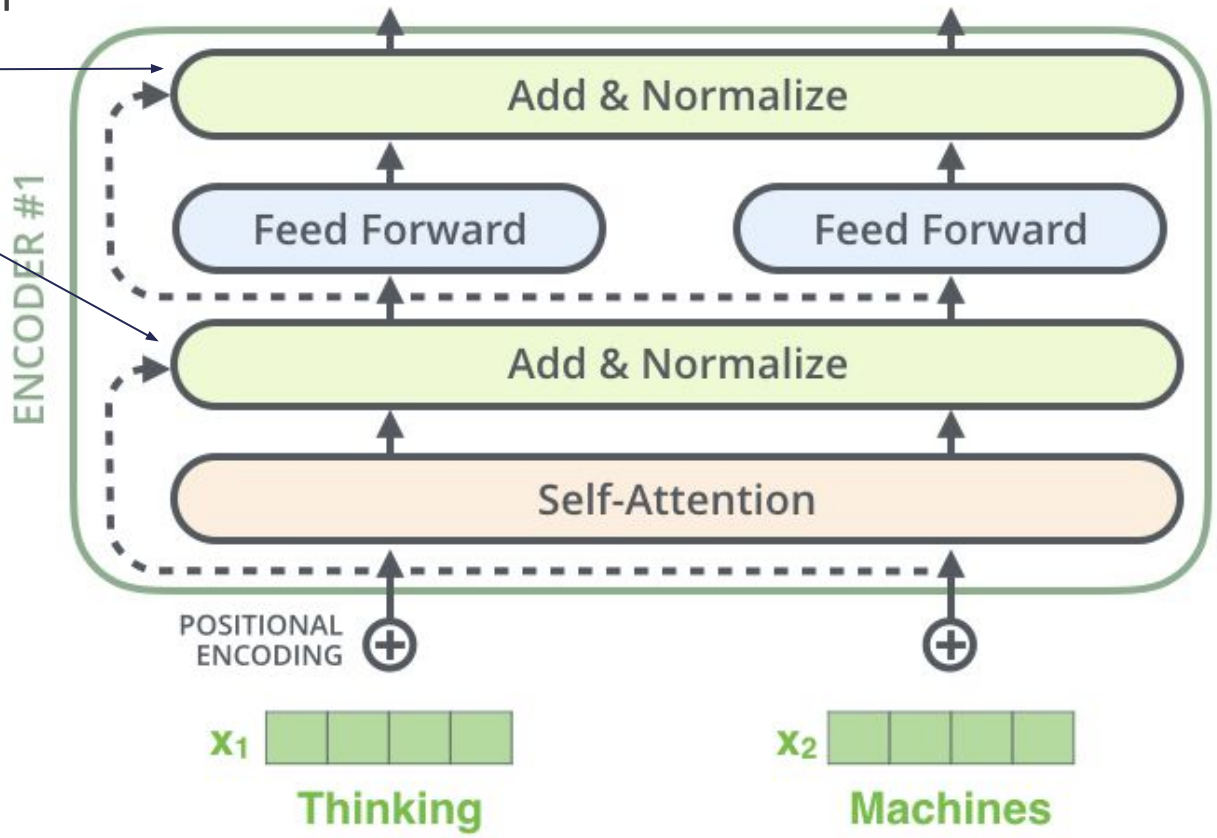


https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Transformer Block

Add & Normalize: propagate information from before the layer, like ResNet

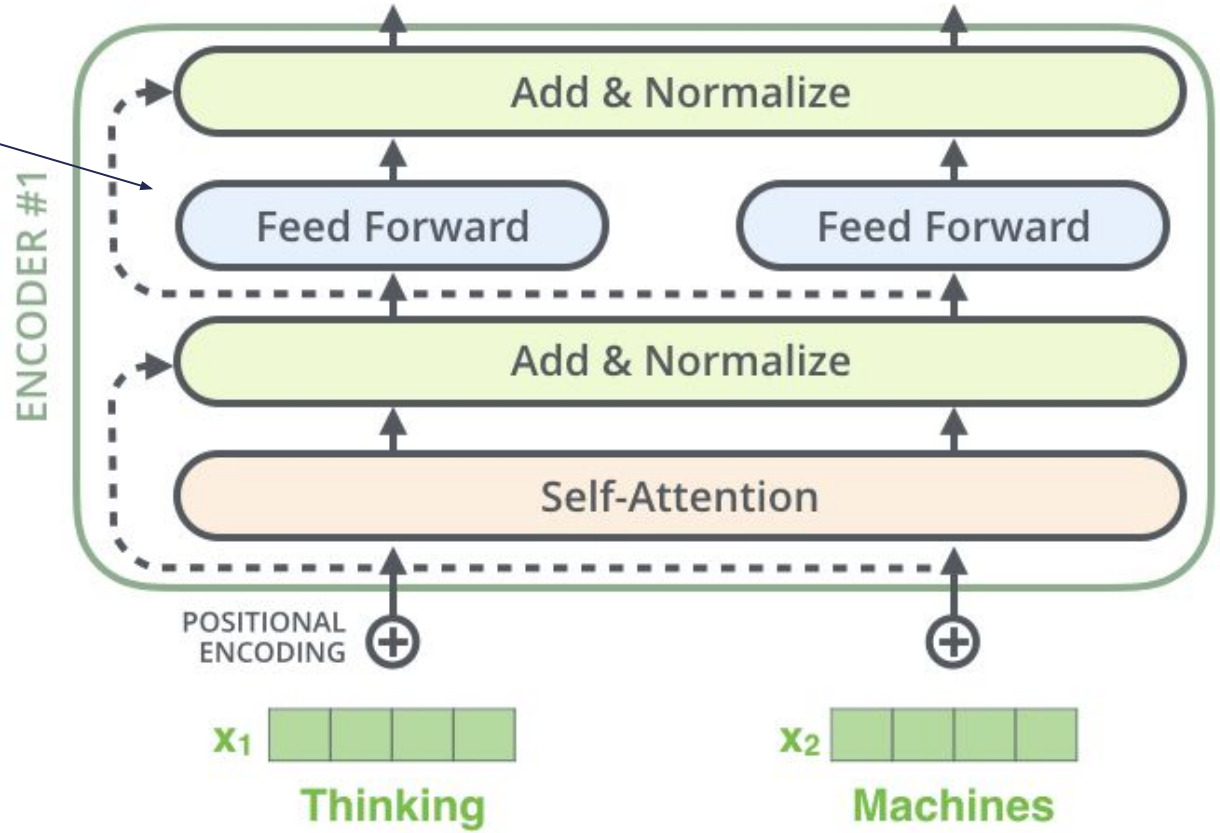
Learn residual transformations, helps gradient flow



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

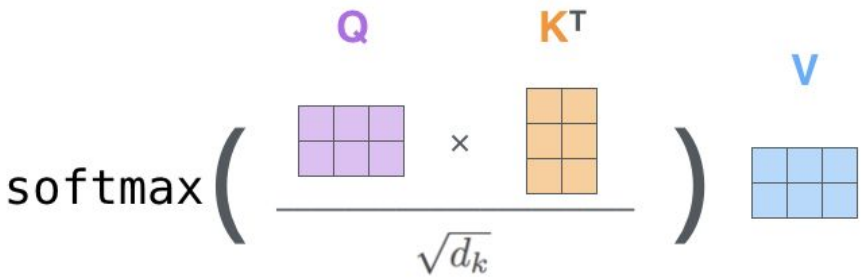
Transformer Block

Feed forward: process the individual tokens
= individual elements



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

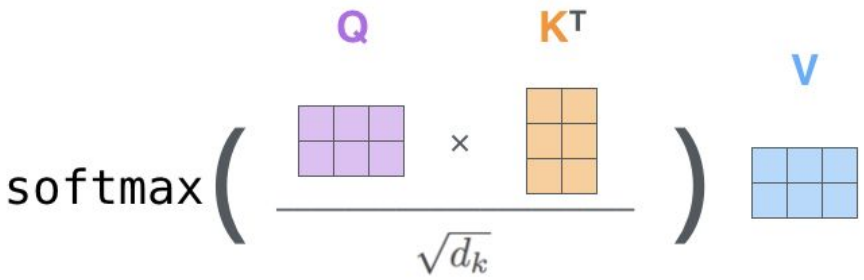
Self-Attention



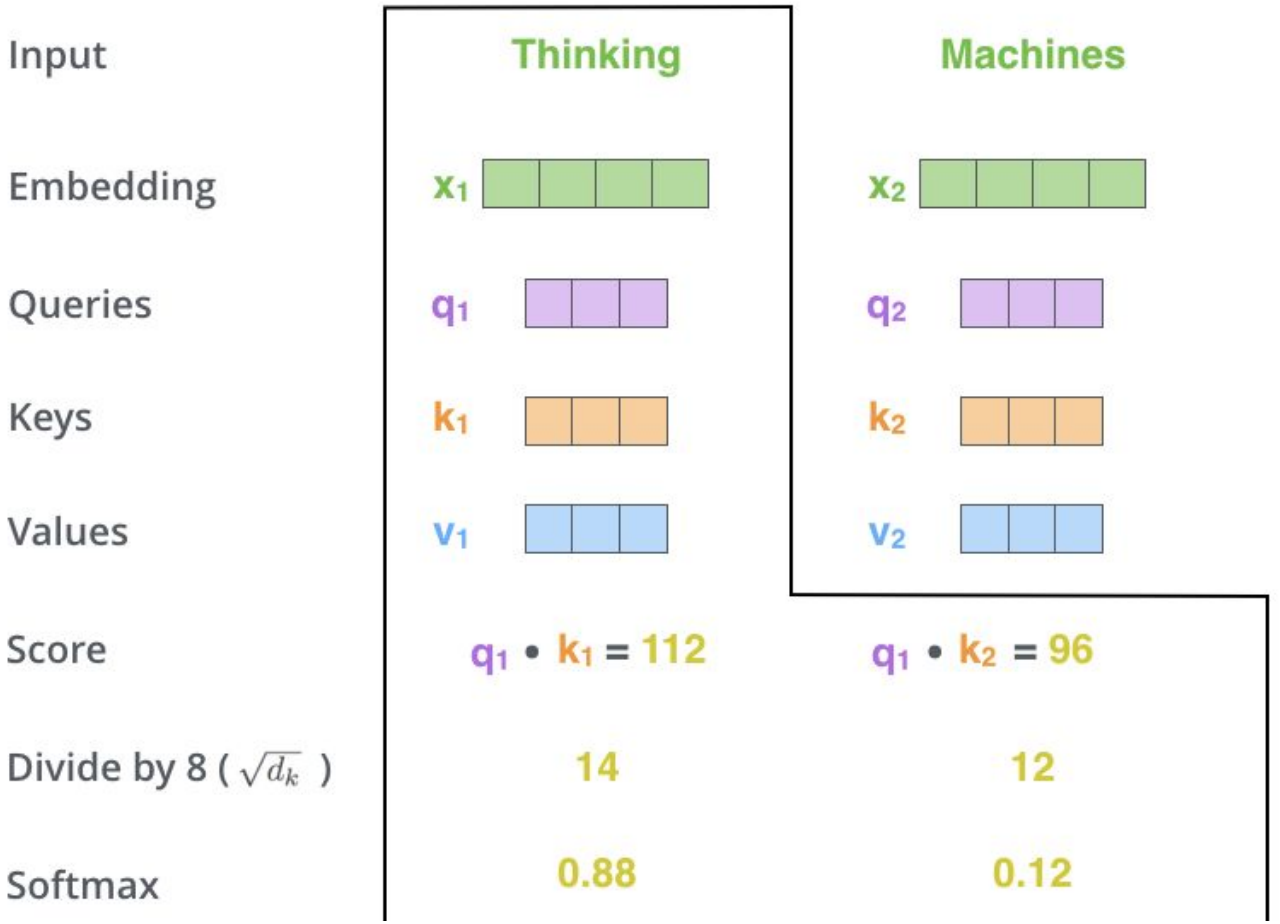
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Self-Attention

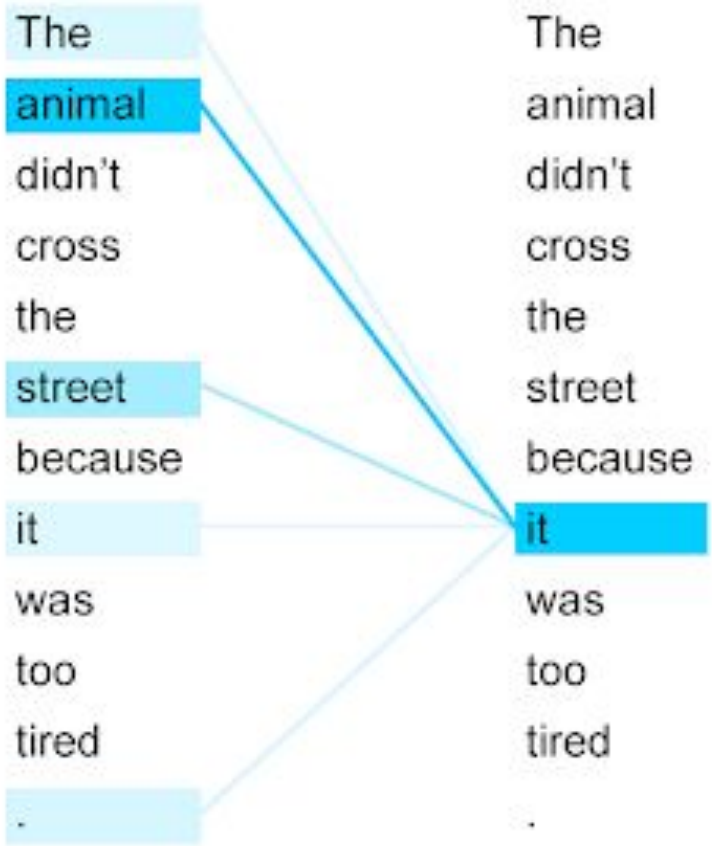


$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Self-Attention



https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Where is the learning happening???

https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Where is the learning happening???

How do we obtain **Q**, **K**, **V**?

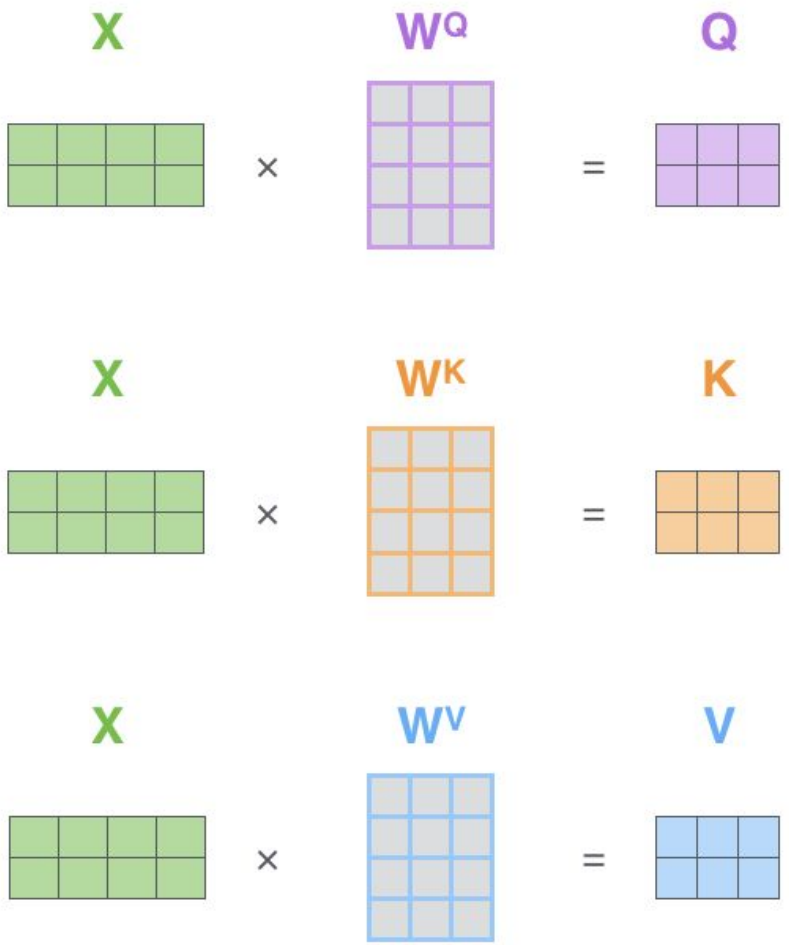
https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Where is the learning happening???

How do we obtain **Q**, **K**, **V**?

Computed from **X** (input tokens) with *learned* weights.

A fully connected layer.



Self-Attention vs. Feed-Forward

Self-Attention:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$Q = XW_q$$
$$K = XW_k$$
$$V = XW_v$$

Fully connected layer:

$$\text{Out} = \text{activation}(W\ X)$$

https://twitter.com/hippopedoid/status/1641432291149848576

Self-Attention vs. Feed-Forward

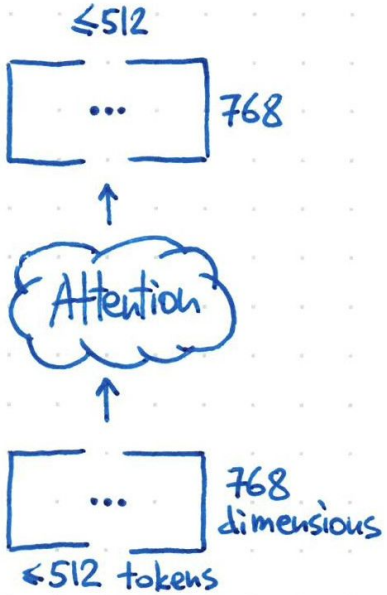
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$Q = XW_q$$

$$K = XW_k$$

$$V = XW_v$$

$$\text{Out} = \text{activation}(W\,X)$$



Self-attention:

- $3 \cdot 768^2$ = 2M params
- works on sequence of any length

Fully-connected:

- $(512 \cdot 768)^2$ = 154 B params
- sequence length (512) is baked in

https://twitter.com/hippopedoid/status/1641432291149848576

Self-Attention vs. Feed-Forward

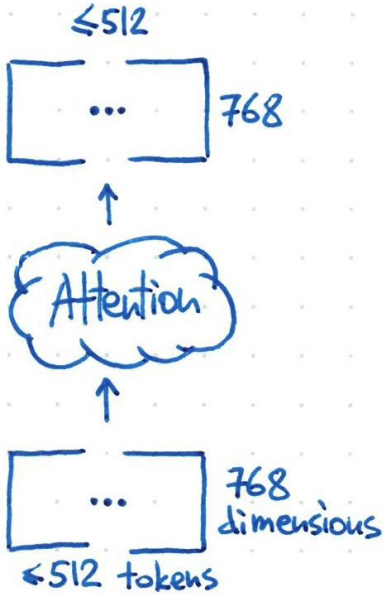
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

$$Q = XW_q$$
$$K = XW_k$$
$$V = XW_v$$

Learnable parameters

$$\text{Out} = \text{activation}(WX)$$

Learnable parameters



Self-attention:

- $3 \cdot 768^2 = 2M$ params
- works on sequence of any length

Fully-connected:

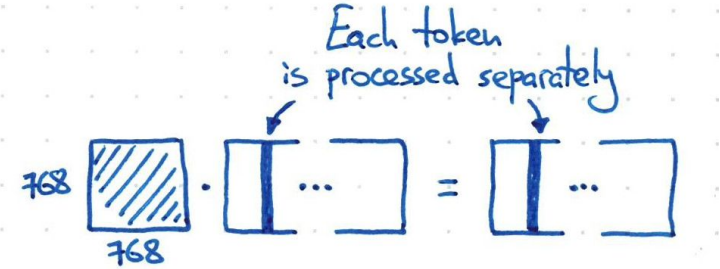
- $(512 \cdot 768)^2 = 154B$ params
- sequence length (512) is baked in

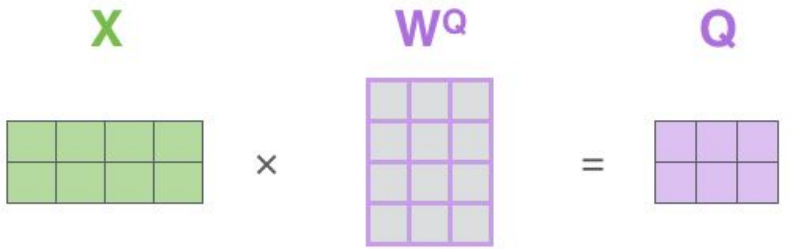
https://twitter.com/hippopedoid/status/1641432291149848576

Self-Attention in a nutshell

The weights:

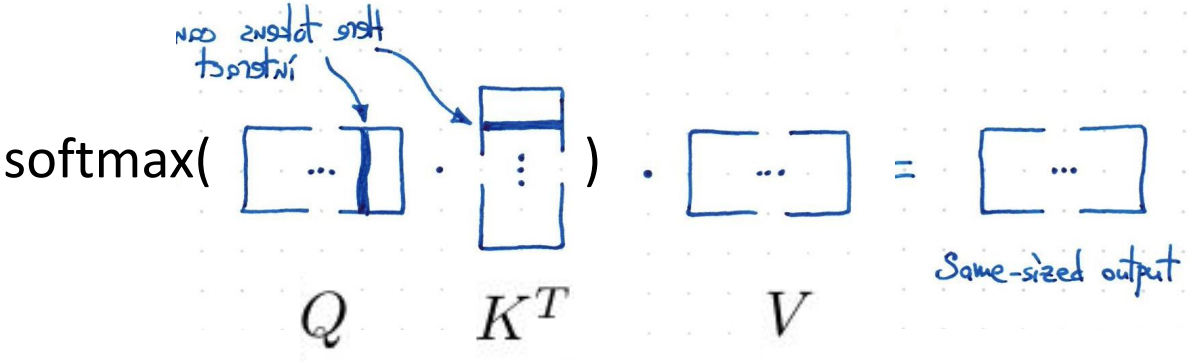
$$W_q \; W_k \; W_v$$





Self-attention:

- arbitrary sequence length
- fixed number of learnable parameters
- encodes pairwise affinity between sequence elements

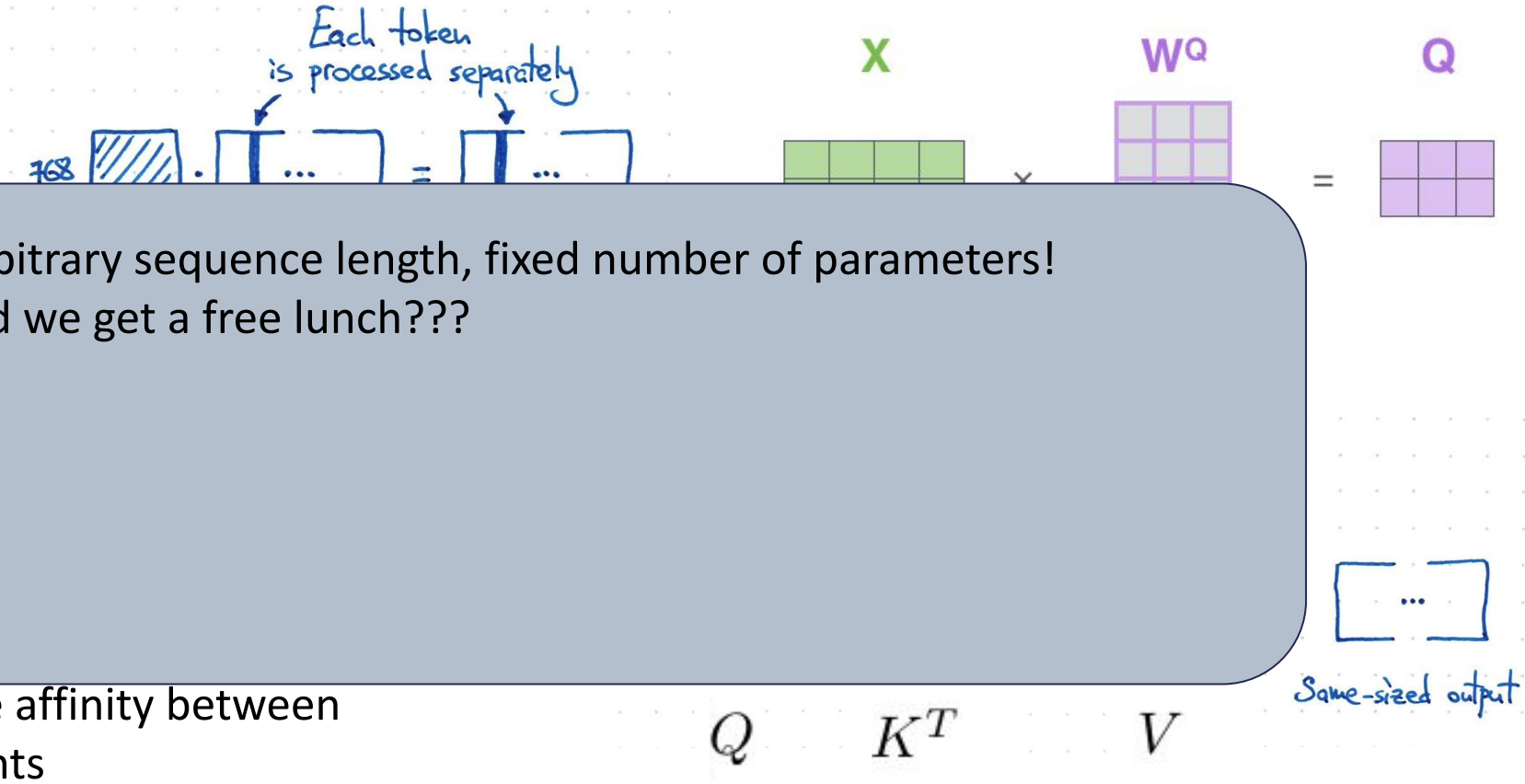


https://twitter.com/hippopedoid/status/1641432291149848576

Self-Attention in a nutshell

The weights:

$W_q \; W_k \; W_v$



Arbitrary sequence length, fixed number of parameters!
Did we get a free lunch???

Self-attention:

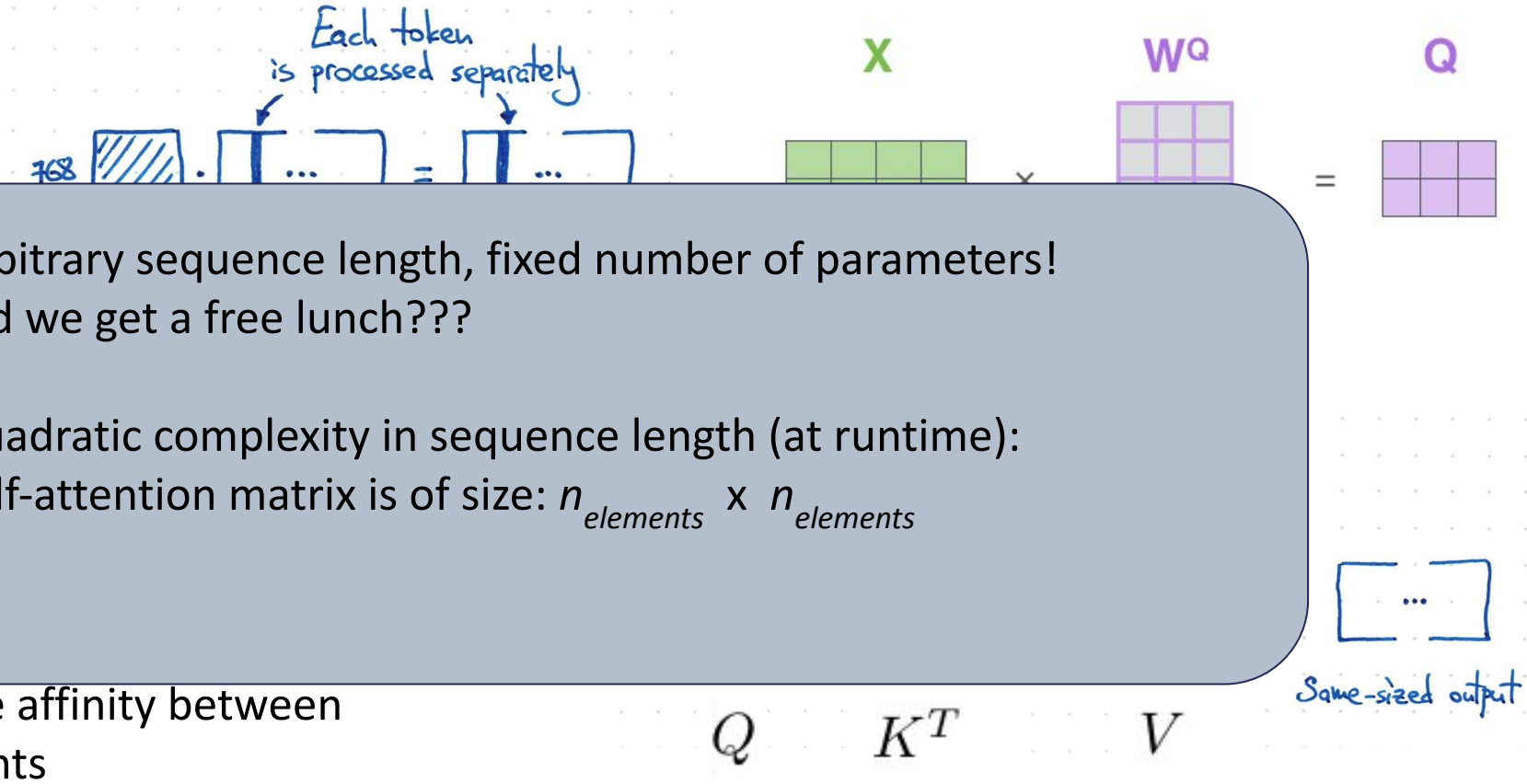
- arbitrary se
- fixed numb
  parameters
- encodes pairwise affinity between sequence elements

https://twitter.com/hippopedoid/status/1641432291149848576

Self-Attention in a nutshell

The weights:

$W_q \; W_k \; W_v$



Arbitrary sequence length, fixed number of parameters!
Did we get a free lunch???

Quadratic complexity in sequence length (at runtime):
Self-attention matrix is of size: $n_{elements}$ x $n_{elements}$

Self-attention:

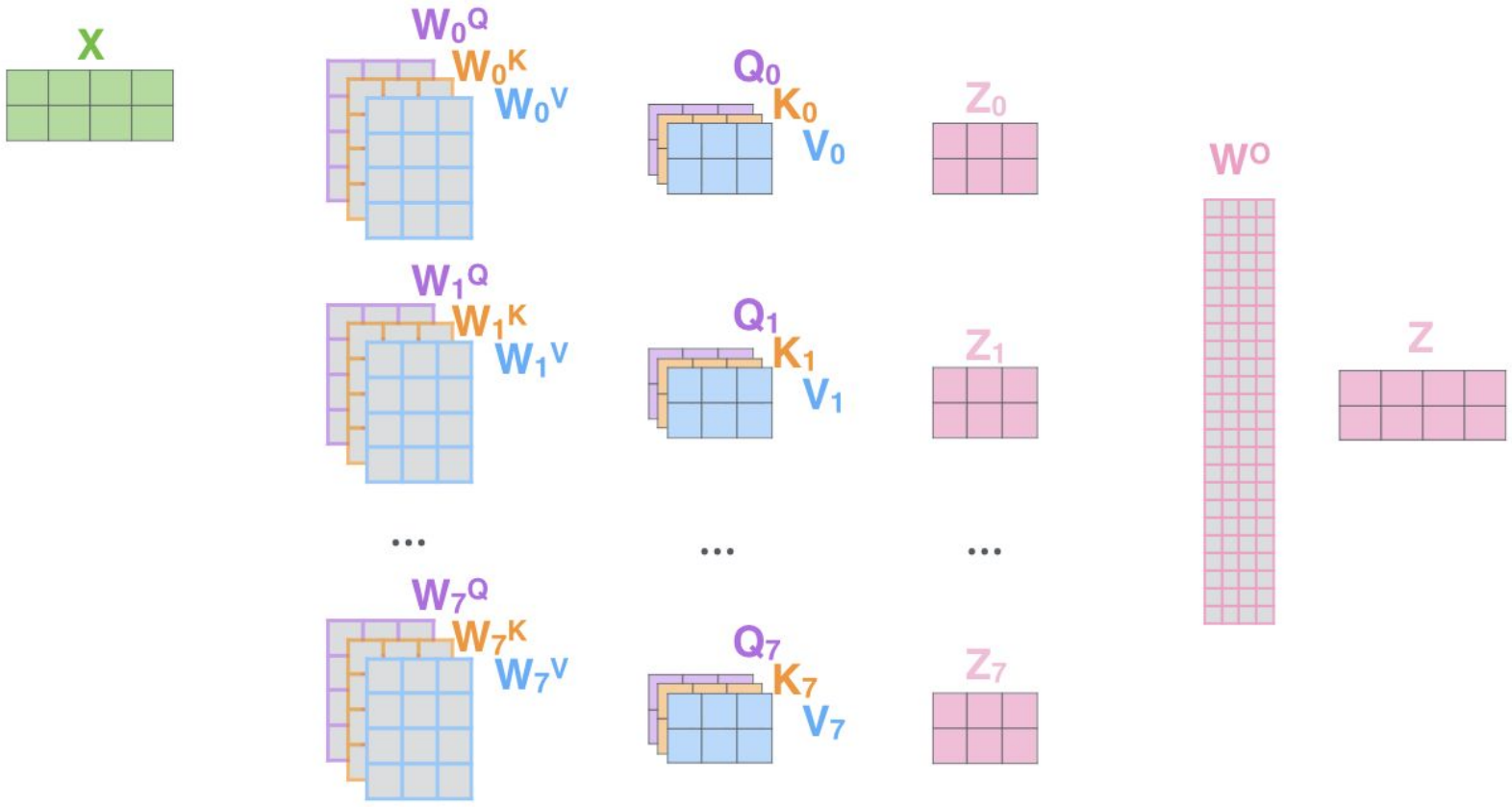
- arbitrary se
- fixed numb
  parameters
- encodes pairwise affinity between sequence elements

https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

Multi-head self-attention

One attention head is limited in learning interactions.

Combine multiple attention heads!

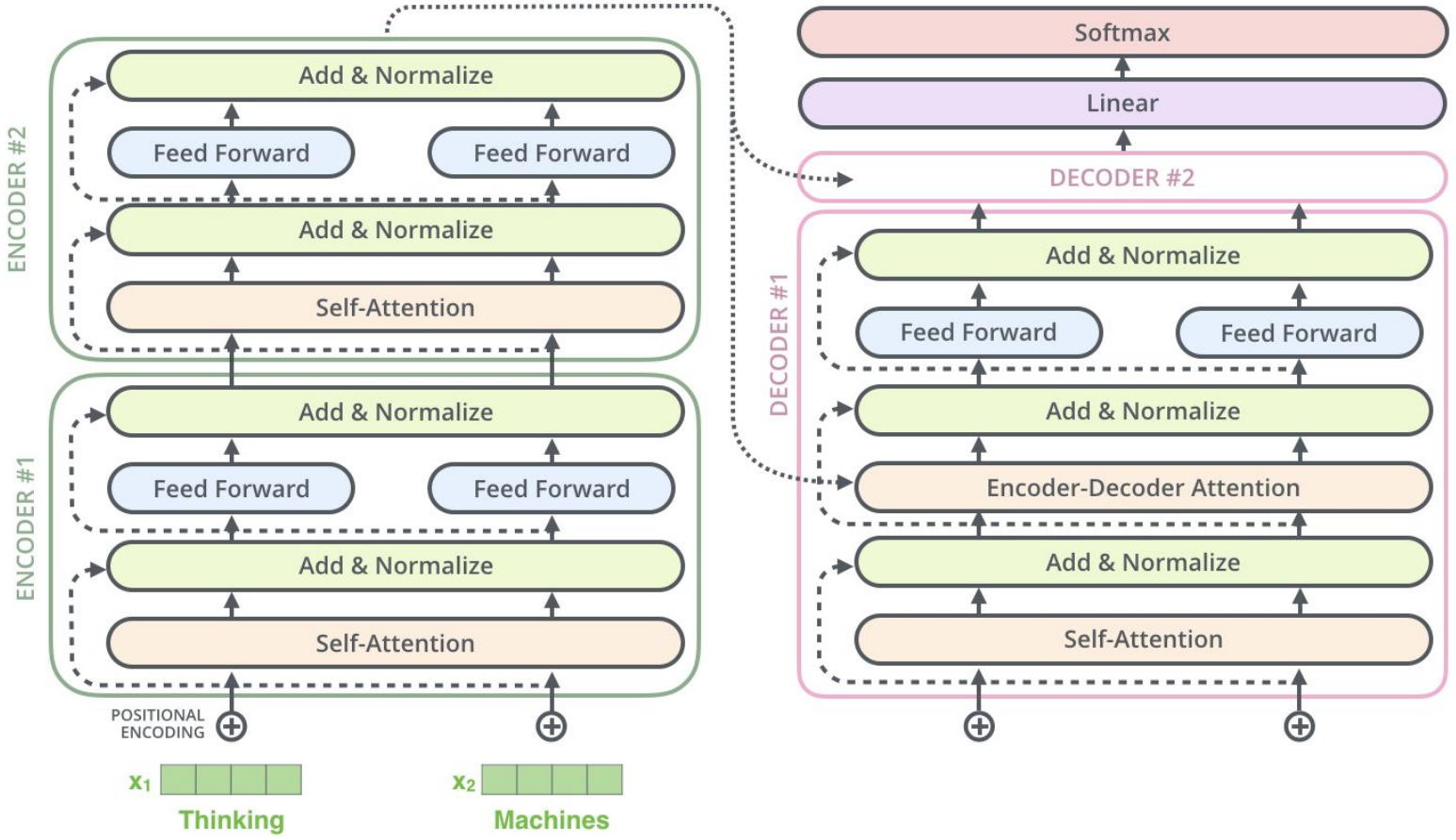


https://jalammar.github.io/illustrated-transformer/
https://arxiv.org/abs/1706.03762

The transformer architecture

Encoder:
stack transformer blocks.
Maps to internal sequence representation

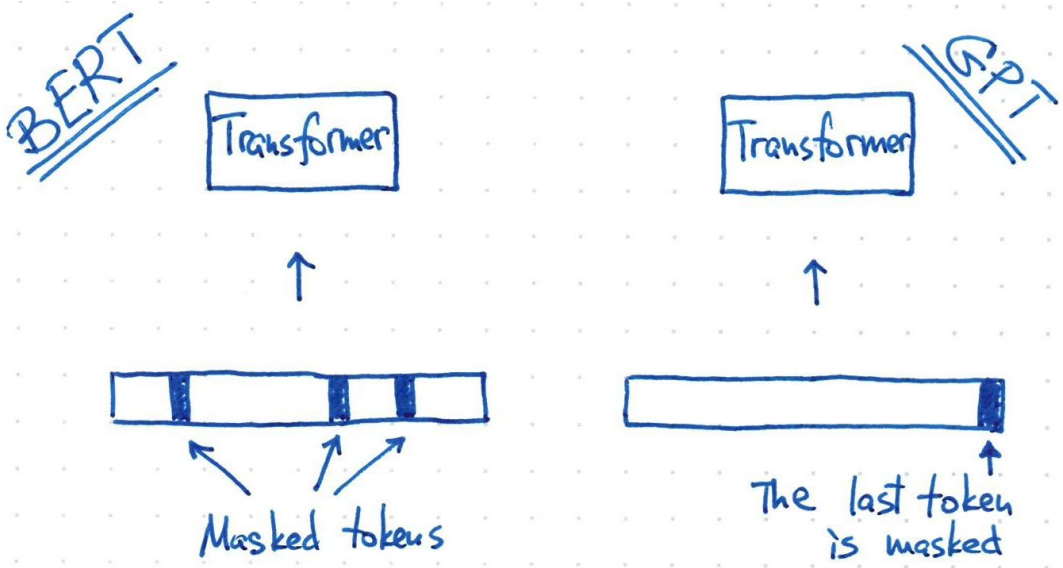
Decoder:
Encoder-Decoder Attention
uses outputs from last layer as keys



BERT: https://arxiv.org/abs/1810.04805
GPT3: https://arxiv.org/abs/2005.14165
Instruction Finetuning: https://arxiv.org/abs/2203.02155
RLHF: https://arxiv.org/abs/1909.08593

Transformers in NLP: Large Language Models

- Autoregressive pretraining: predict part of sequence from the rest, trained on large corpus
  - BERT: Mask random tokens
  - GPT: Mask last token
  - ChatGPT, GPT4, LLAMA, …



- Application to downstream tasks (Translation, Chatbot, …)
  - Instruction fine-tuning (training on supervised samples)
  - Reinforcement learning from human feedback (RLHF) (rating of reponses)

Vision transformers

Why and how?

Motivations:

- Success in NLP
- “Scaling laws”: transformer performance scales with dataset size
- Sequence representation is universal -> combine vision, text etc.
- Remove inductive biases?

Why and how?

Motivations:

- Success in NLP
- “Scaling laws”: transformer performance scales with dataset size
- Sequence representation is universal -> combine vision, text etc.
- Remove inductive biases?

How can we represent an image as a sequence?

Why and how?

Motivations:

- Success in NLP
- “Scaling laws”: transformer performance scales with dataset size
- Sequence representation is universal -> combine vision, text etc.
- Remove inductive biases?

How can we represent an image as a sequence?

- Tokenize every pixel: too many elements! (Remember quadratic complexity)

- Tokenize small patches!

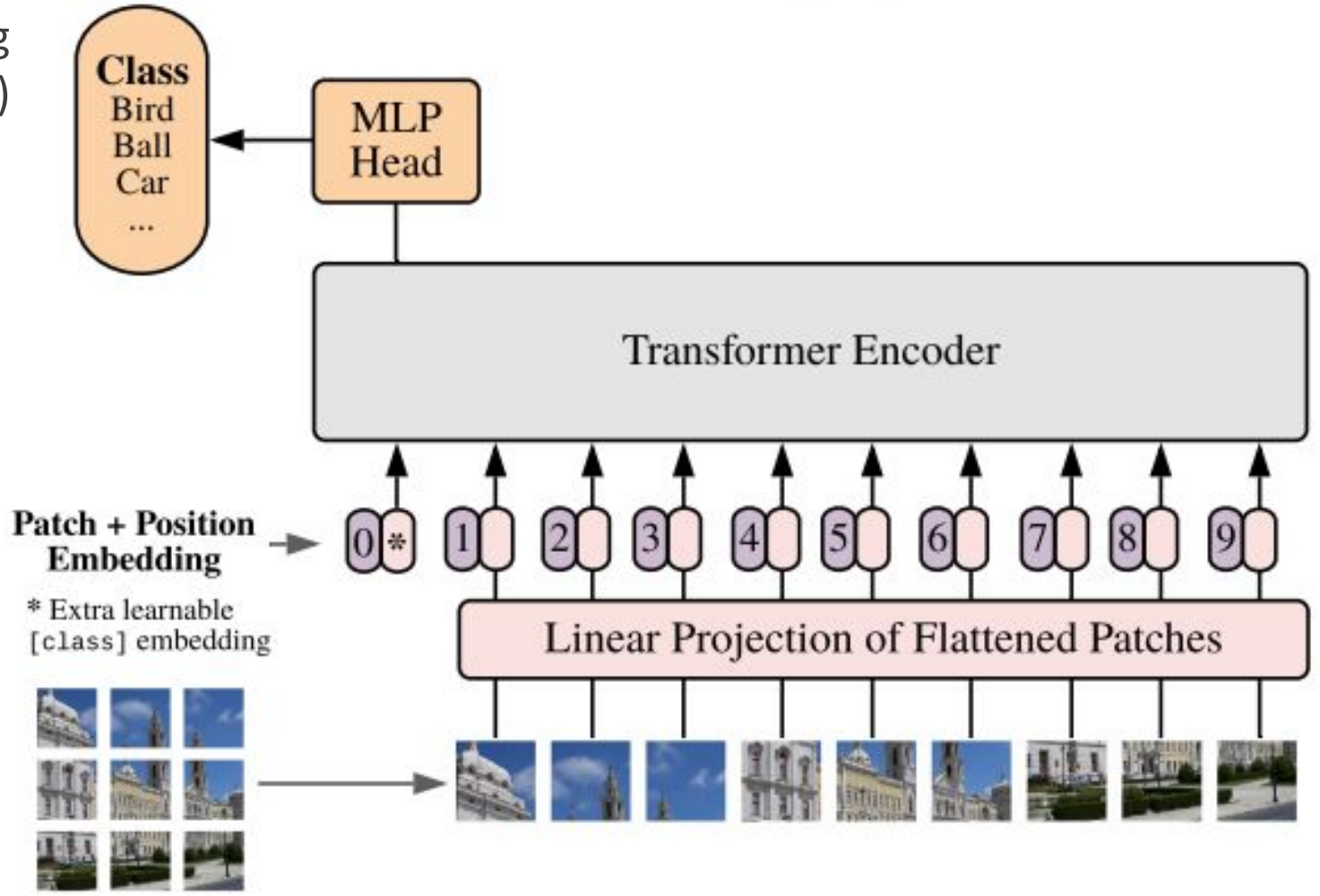
https://arxiv.org/abs/2010.11929

The Vision Transformer (ViT)

Tokenize patches + position embedding (encode position of patch in the image)

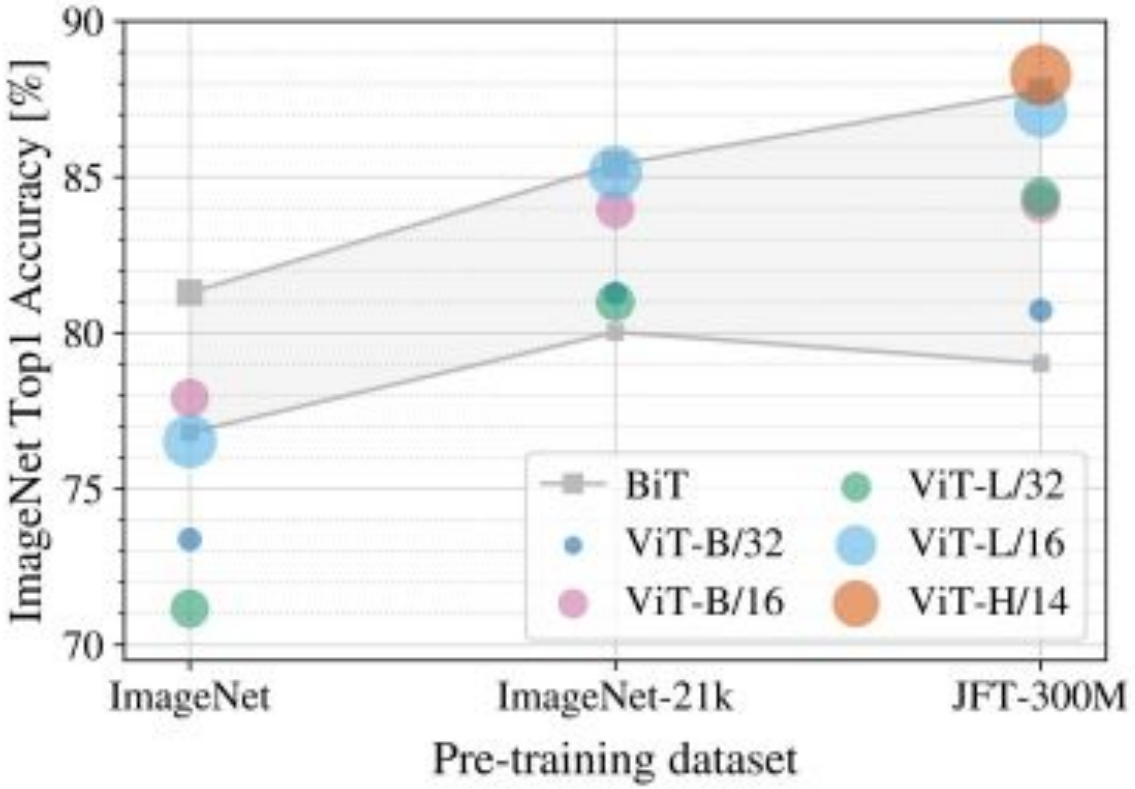
Encoder: same as in NLP

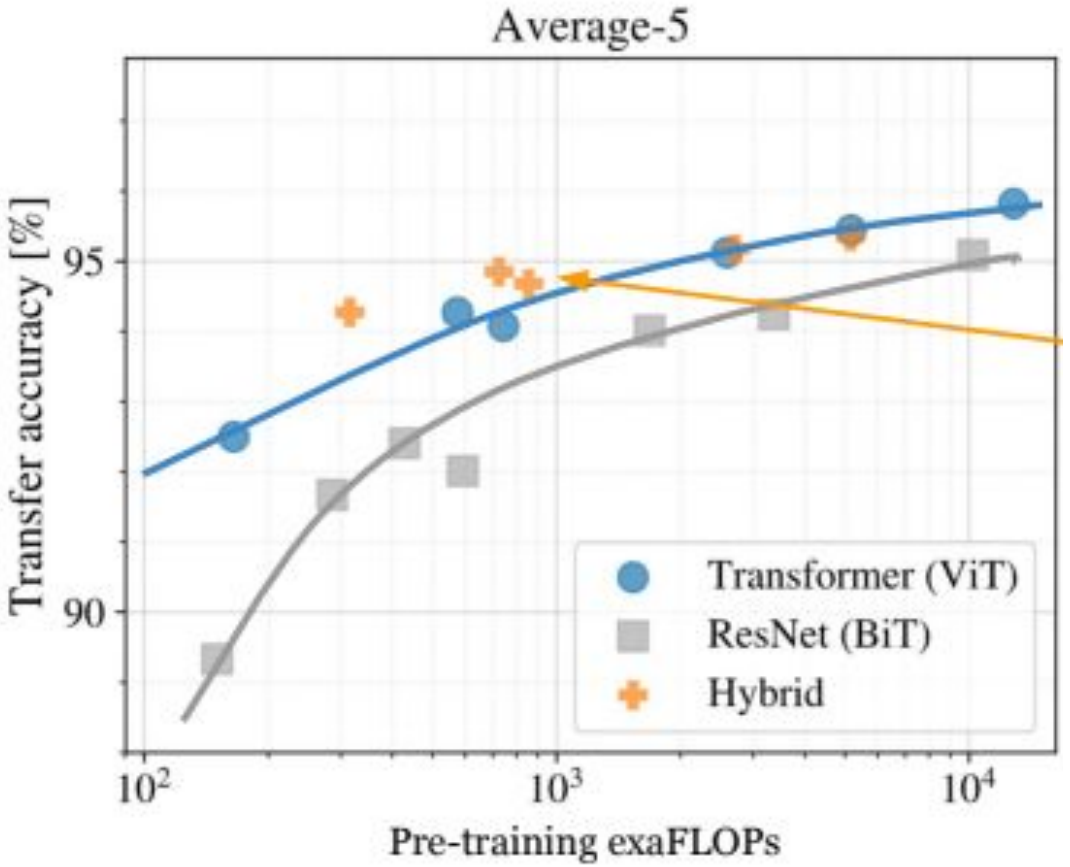
Classification based head on top of transformer output.



https://arxiv.org/abs/2010.11929

The Vision Transformer (ViT)



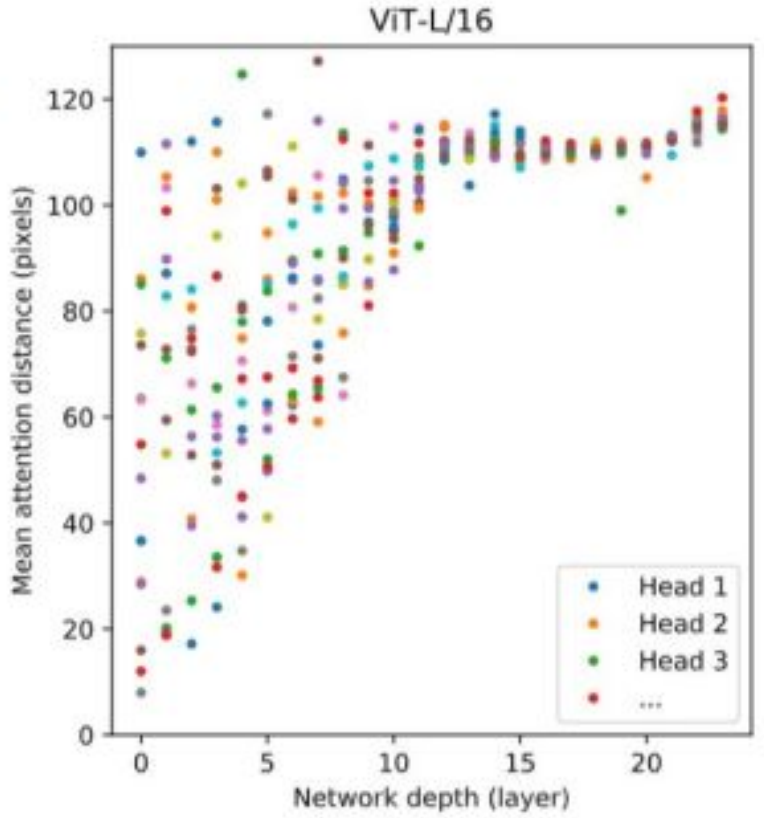


https://sites.google.com/view/cvpr-2022-beyond-cnn
https://arxiv.org/abs/2010.11929

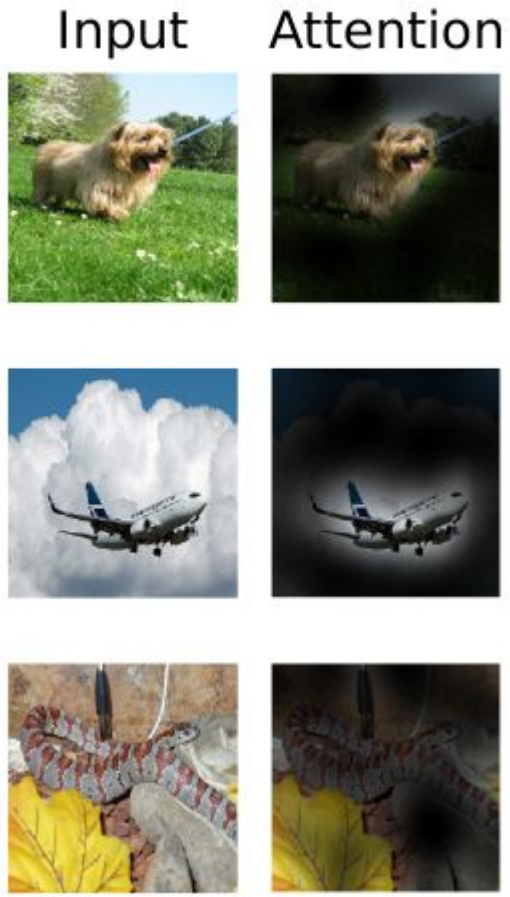
What does attention learn?

Spatially global features

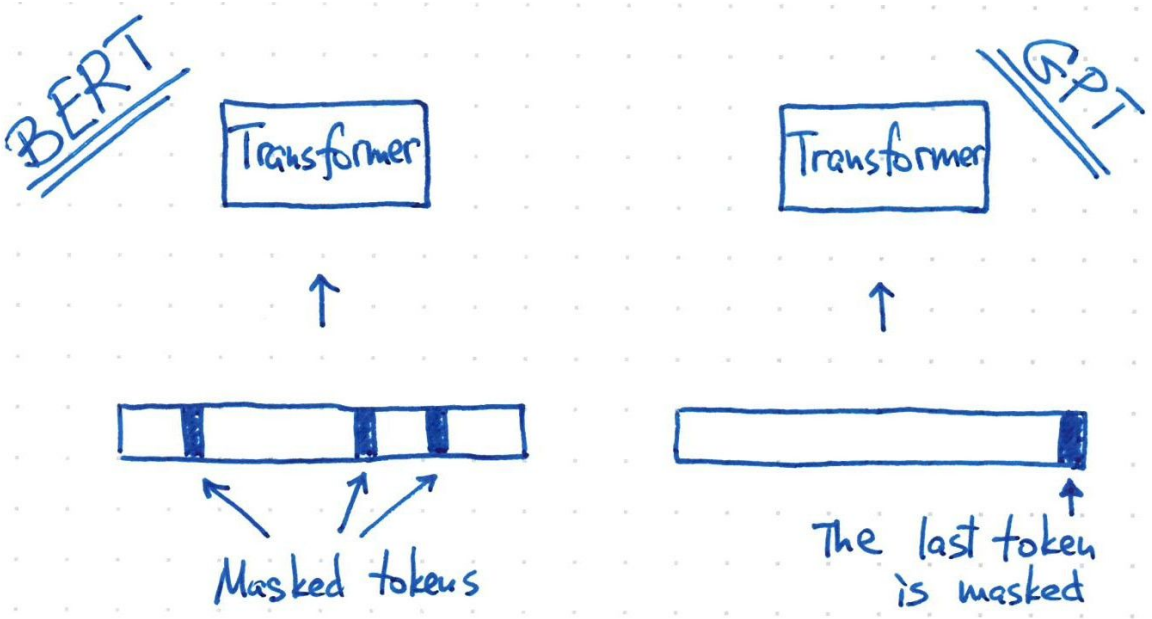
Spatially local features



Inspection of learned attention heads



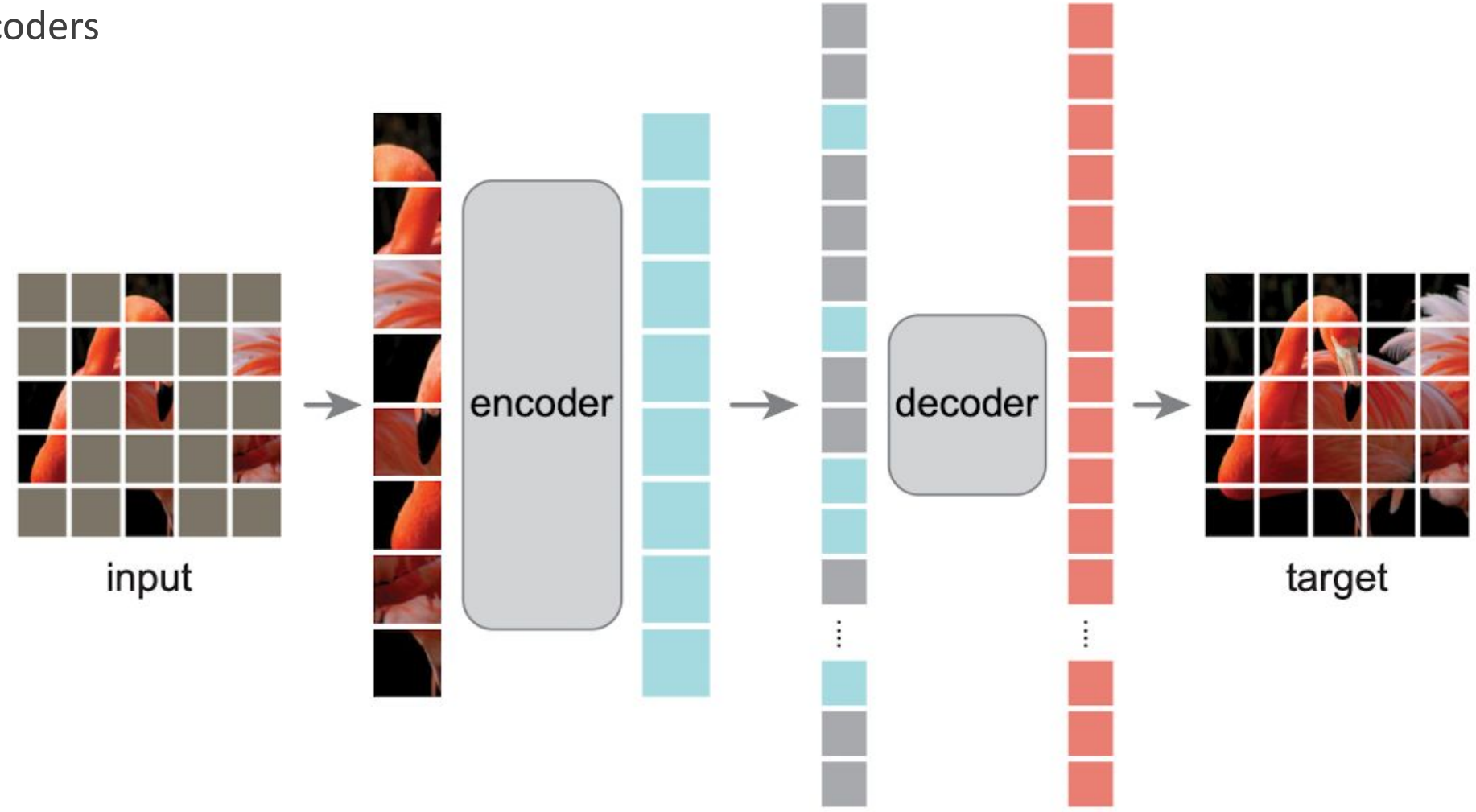
Autoregressive training with ViT?



https://arxiv.org/abs/2111.06377

Autoregressive training with ViT!

Masked Auto Encoders



Vision Foundation Models

- CLIP: Transformers that encode text and image to shared representation
  - Main ingredient of generative image methods: DALL-E, Stable Diffusion, …
- DINO v2: Large vision transformer trained with different self-supervised losses
  - Powerful image features that enable many downstream tasks
- GPT4 Vision / Gemini Vision / …
  - Presumably similar architecture to CLIP (no publication!)
- **Segment Anything Model**
  - Towards universal and promptable segmentation

Segment Anything

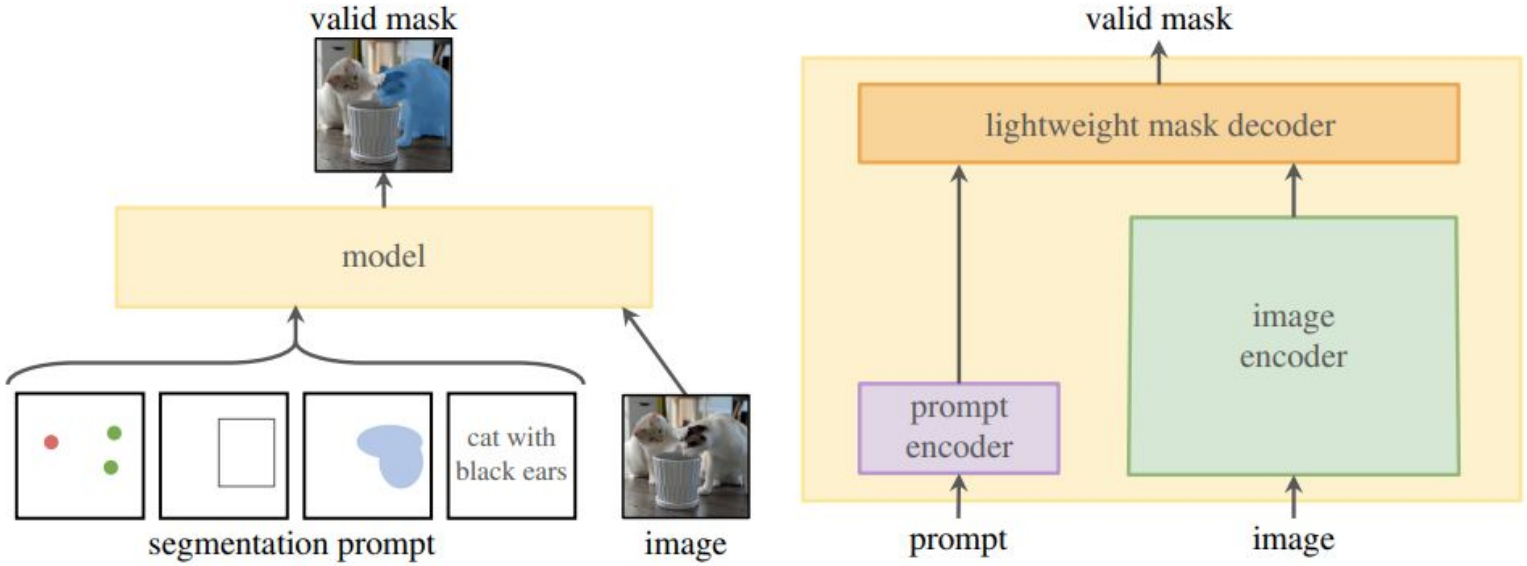
Vision foundation model for segmentation

Segment Anything

https://arxiv.org/abs/2304.02643

Pretrained model for interactive segmentation from Meta.AI

SAM: Interactive segmentation

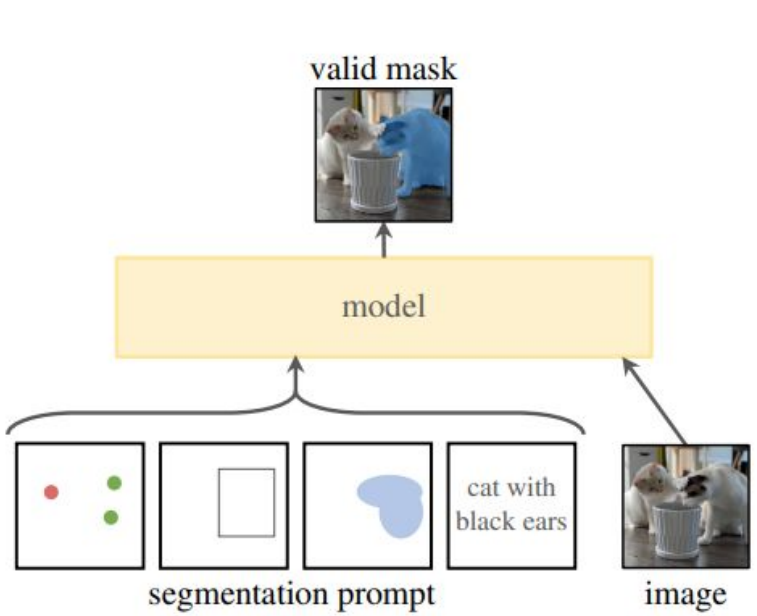


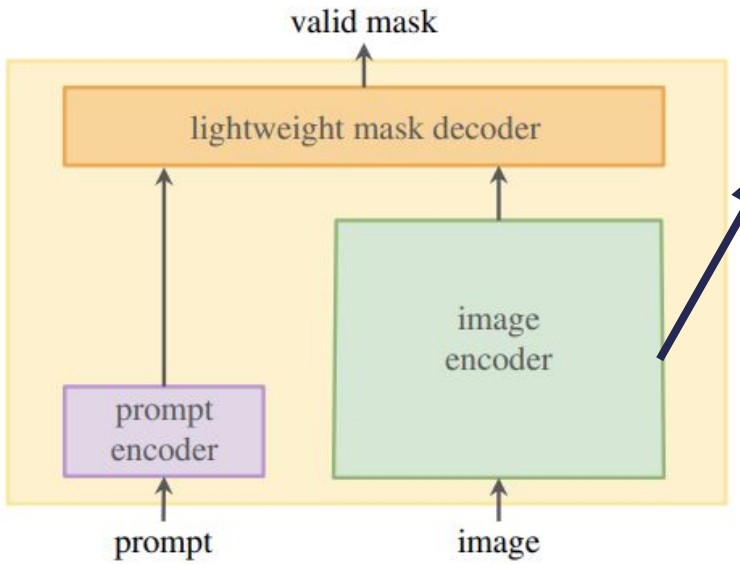
Segment Anything

https://arxiv.org/abs/2304.02643
* MobileSAM:
https://arxiv.org/abs/2306.14289

Pretrained model for interactive segmentation from Meta.AI

SAM: Interactive segmentation





4 different sizes:
- VIT-B (Base)
- VIT-L (Large)
- VIT-H (Huge)
- VIT-T (Tiny)*

Segment Anything: What's special?

- Interactive segmentation: segment (almost) arbitrary objects from annotations
  - "prompts": points and/or box and/or mask
  - more prompts improve the predictions

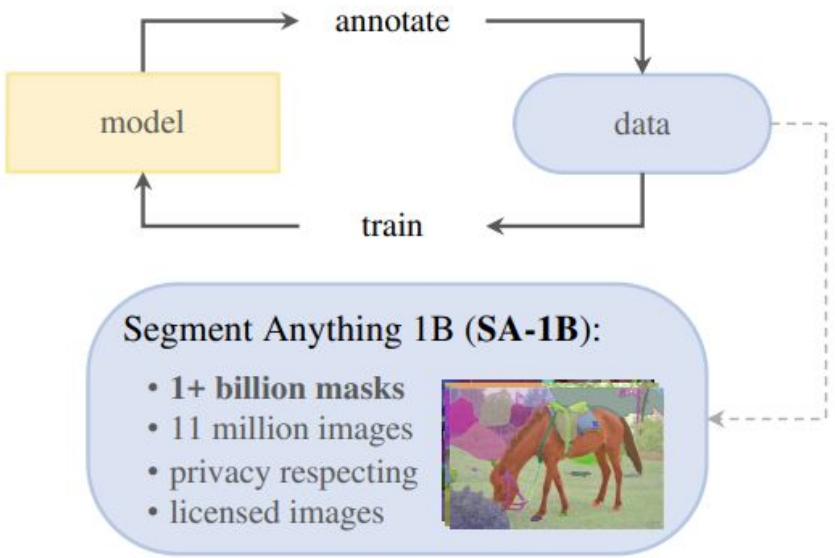
- Versatile: can be integrated within pipelines that provide prompts
  - From user inputs, object detectors, nucleus seeds, …
  - Model is fully open-source!

Segment Anything: What's special?

- Interactive segmentation: segment (almost) arbitrary objects from annotations
  - "prompts": points and/or box and/or mask
  - more prompts improve the predictions

- Versatile: can be integrated within pipelines that provide prompts
  - From user inputs, object detectors, nucleus seeds, …
  - Model is fully open-source!

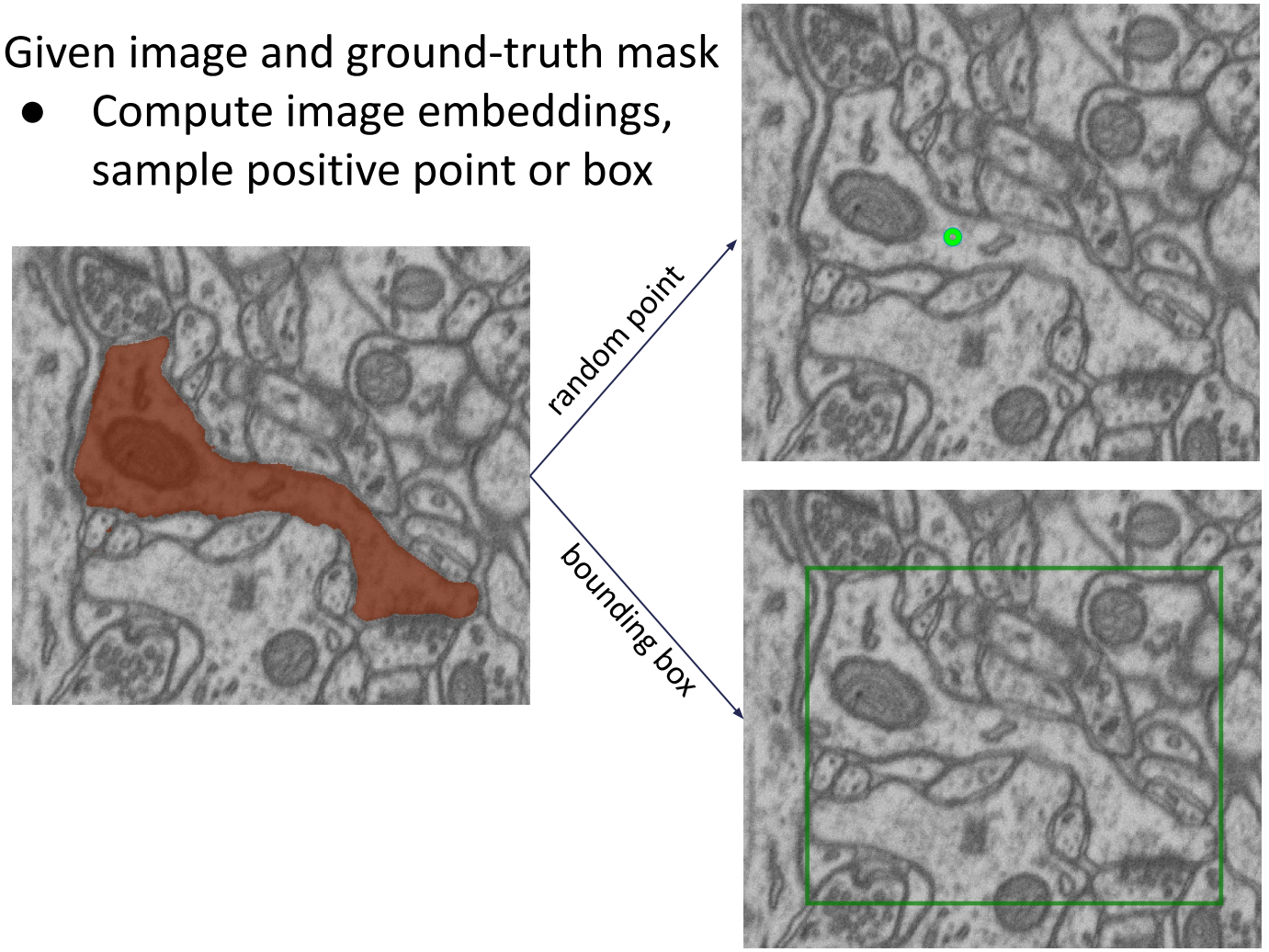
- **How?**
  - **Large dataset with diverse images and objects**
  - Iterative training loop



Segment Anything: Training iteration

Given image and ground-truth mask

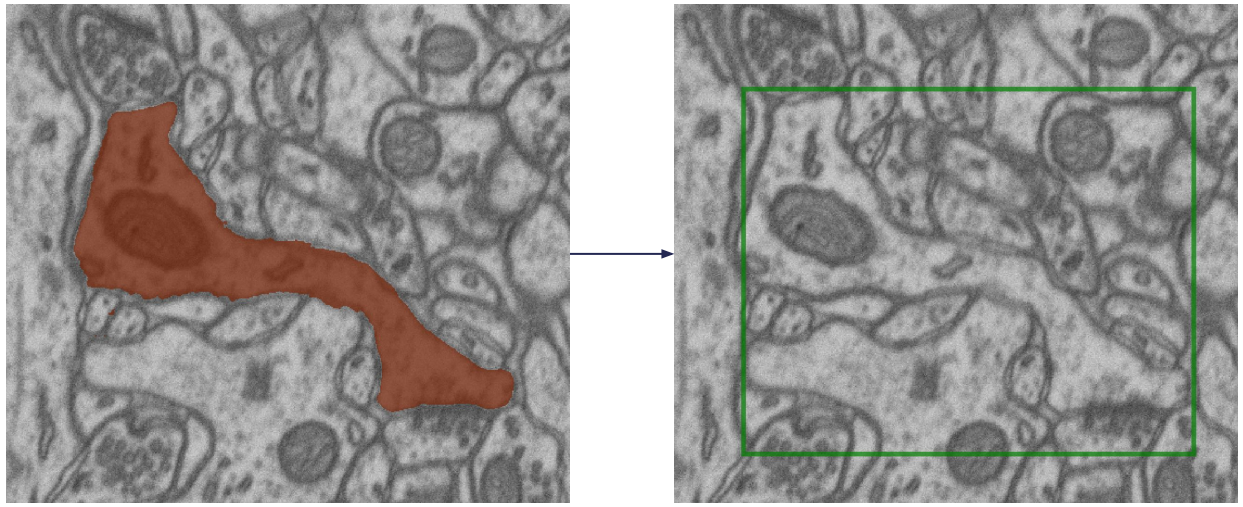
- Compute image embeddings, sample positive point or box



Segment Anything: Training iteration

Given image and ground-truth mask

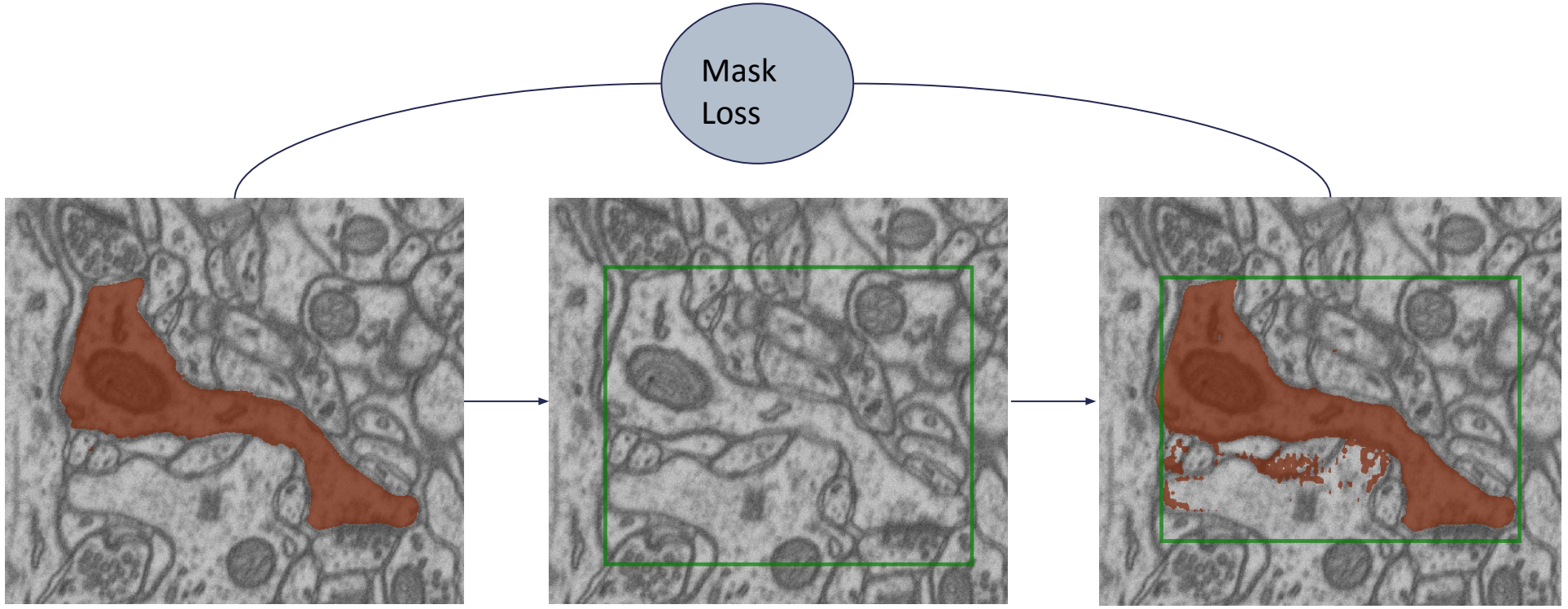
- Compute image embeddings, sample positive point or **box**



Segment Anything: Training iteration

Given image and ground-truth mask

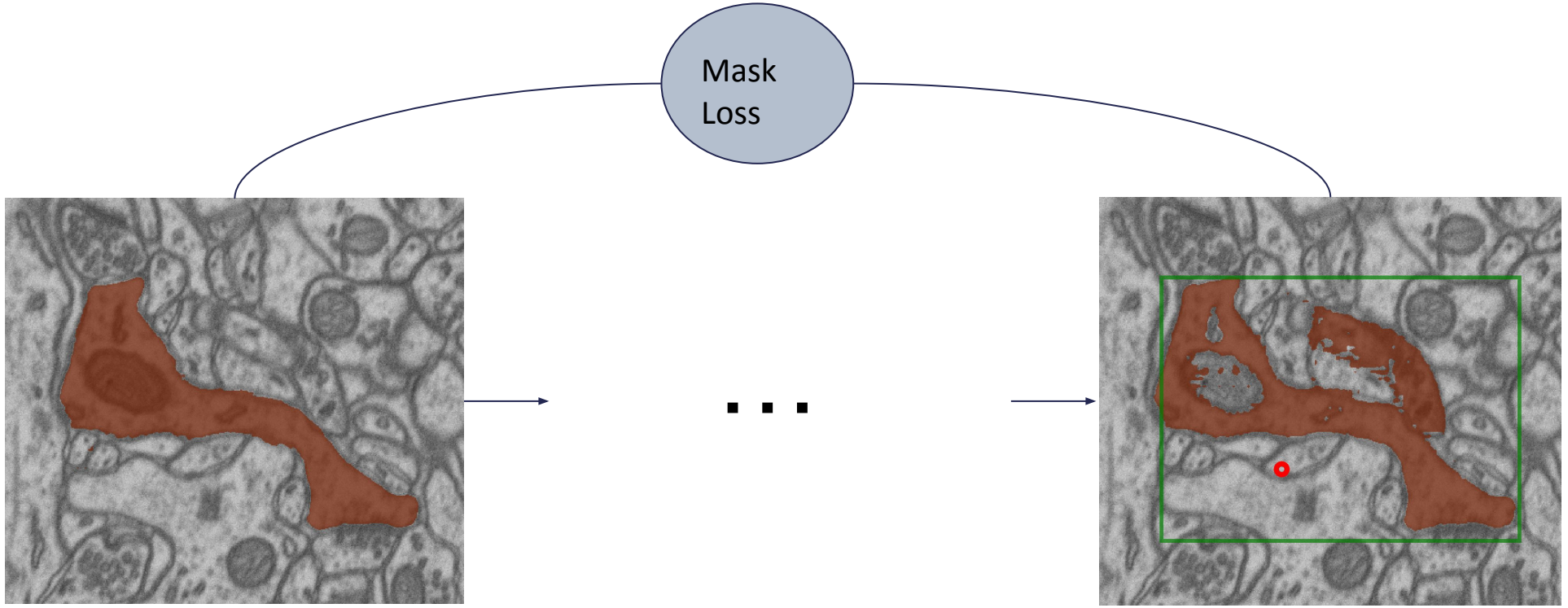
- Compute image embeddings, sample positive point or box
- Run prediction, compute loss for object and IOU estimate



Segment Anything: Training iteration

Given image and ground-truth mask

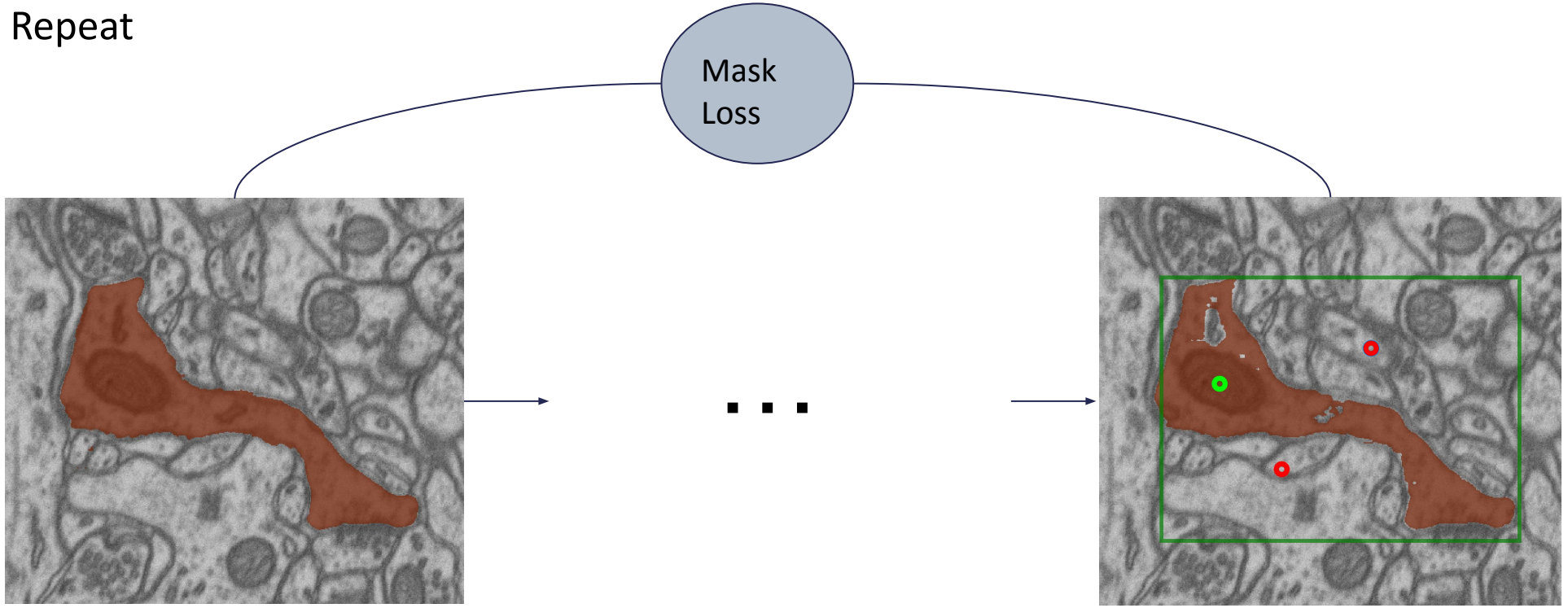
- Compute image embeddings, sample positive point or box
- Run prediction, compute loss for object and IOU estimate
- Sample point prompts where prediction is wrong, rerun prediction with all prompts + mask



Segment Anything: Training iteration

Given image and ground-truth mask

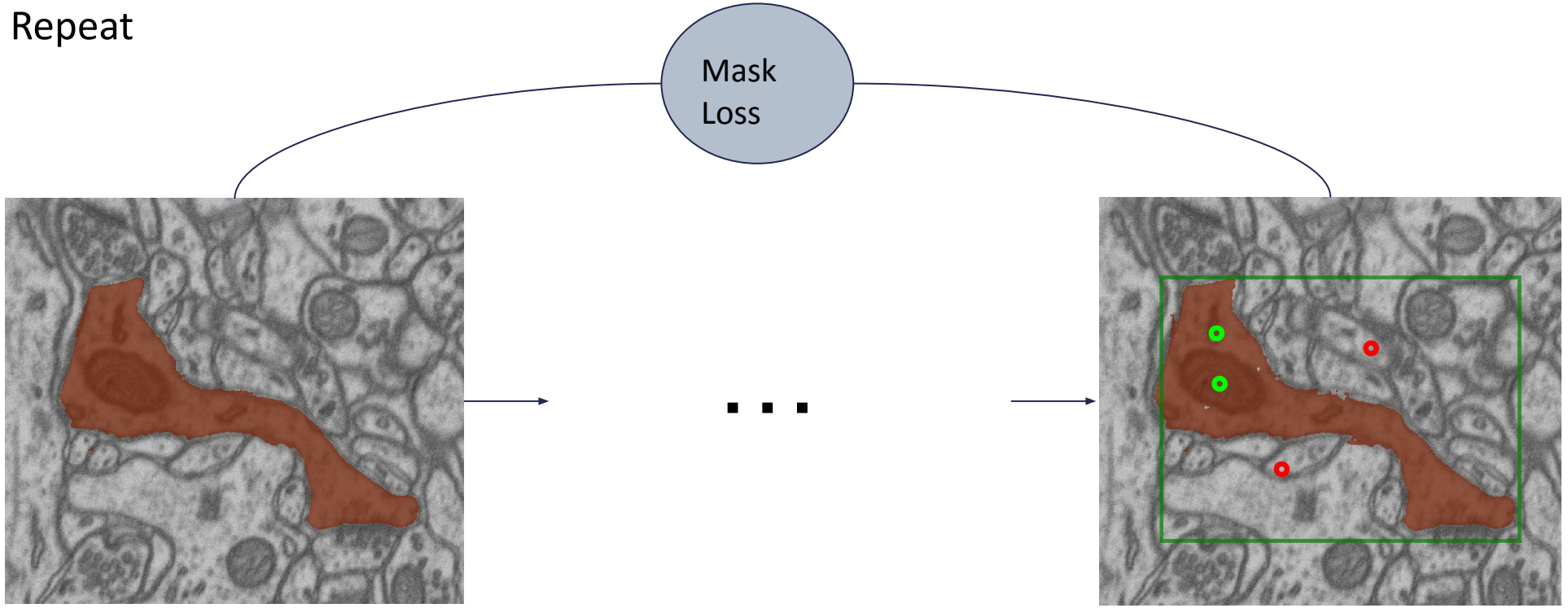
- Compute image embeddings, sample positive point or box
- Run prediction, compute loss for object and IOU estimate
- Sample point prompts where prediction is wrong, rerun prediction with all prompts + mask
- Repeat



Segment Anything: Training iteration

Given image and ground-truth mask

- Compute image embeddings, sample positive point or box
- Run prediction, compute loss for object and IOU estimate
- Sample point prompts where prediction is wrong, rerun prediction with all prompts + mask
- Repeat



Segment Anything: Training iteration

Given image and ground-truth mask

- Compute image embeddings, sample positive point or box
- Run prediction, compute loss for object and IOU estimate
- Sample point prompts where prediction is wrong, rerun prediction with all prompts + mask
- Repeat
- Average losses, update weights

Segment Anything: Capabilities

https://segment-anything.com/

Segmentation from user inputs (prompts)

Segment Anything: Capabilities

https://segment-anything.com/

Segmentation from user inputs (prompts)

Automatic Mask Generation (AMG)

Segment Anything for Microscopy

Anwai
Archit

Our aims & contributions

Archit, …, **Pape**, *bioRxiv (2023)*
https://doi.org/10.1101/2023.08.21.554208

- How well does SAM work for microscopy data? Which model size is best?
- Can we improve it (by finetuning) on microscopy data?
- Build a napari-based tool for interactive and automatic segmentation and tracking.

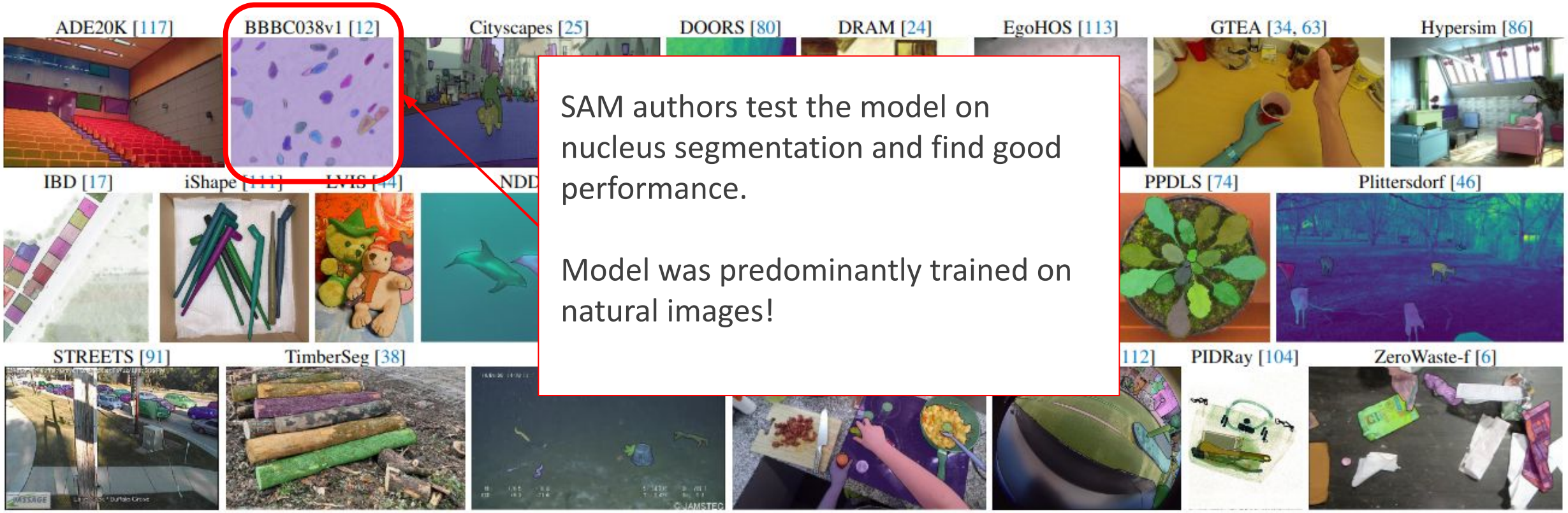
Collaboration between my group and DFKI; + several open source contributions.

Anwai
Archit

Our aims & contributions

Archit, …, **Pape**, *bioRxiv (2023)*
https://doi.org/10.1101/2023.08.21.554208

- How well does SAM work for microscopy data? Which model size is best?



SAM authors test the model on nucleus segmentation and find good performance.

Model was predominantly trained on natural images!

Anwai
Archit

Our aims & contributions

Archit, …, **Pape**, *bioRxiv (2023)*
https://doi.org/10.1101/2023.08.21.554208

- How well does SAM work for microscopy data? Which model size is best?
- Can we improve it by finetuning on microscopy data?
- Build a napari-based tool for interactive and automatic segmentation and tracking.

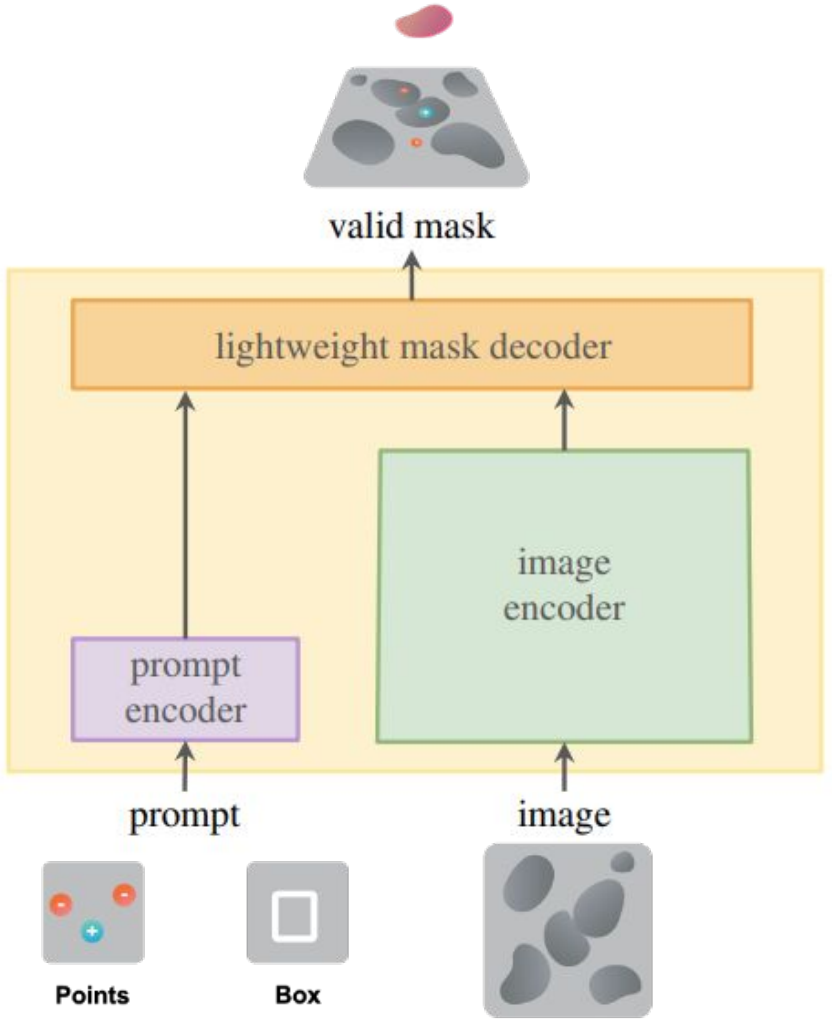
We just released our stable v1.0 version!

And submitted revision -> experiments are not in our preprint yet.

Finetuning SAM

Our contributions:

- Re-implement iterative training
  - Original code not published
  - Complex procedure
  - Use to finetune **SAM components**

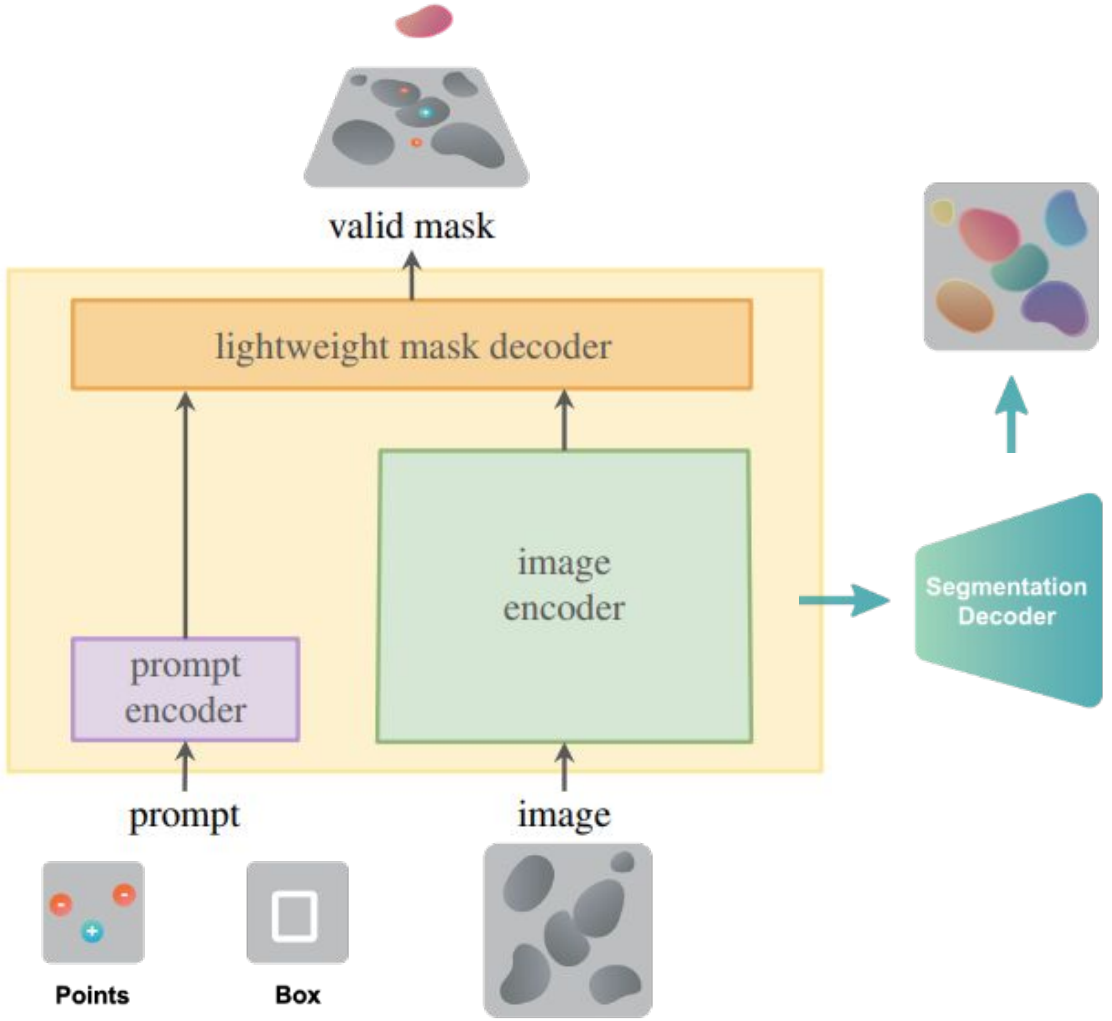


Finetuning SAM + improve instance seg

Our contributions:

- Re-implement iterative training
  - Original code not published
  - Complex procedure
  - Use to finetune SAM components

- Add decoder for instance segmentation (AIS)
  - Predicts foreground
  - Regresses distances to boundary + centroid
  - Input for watershed



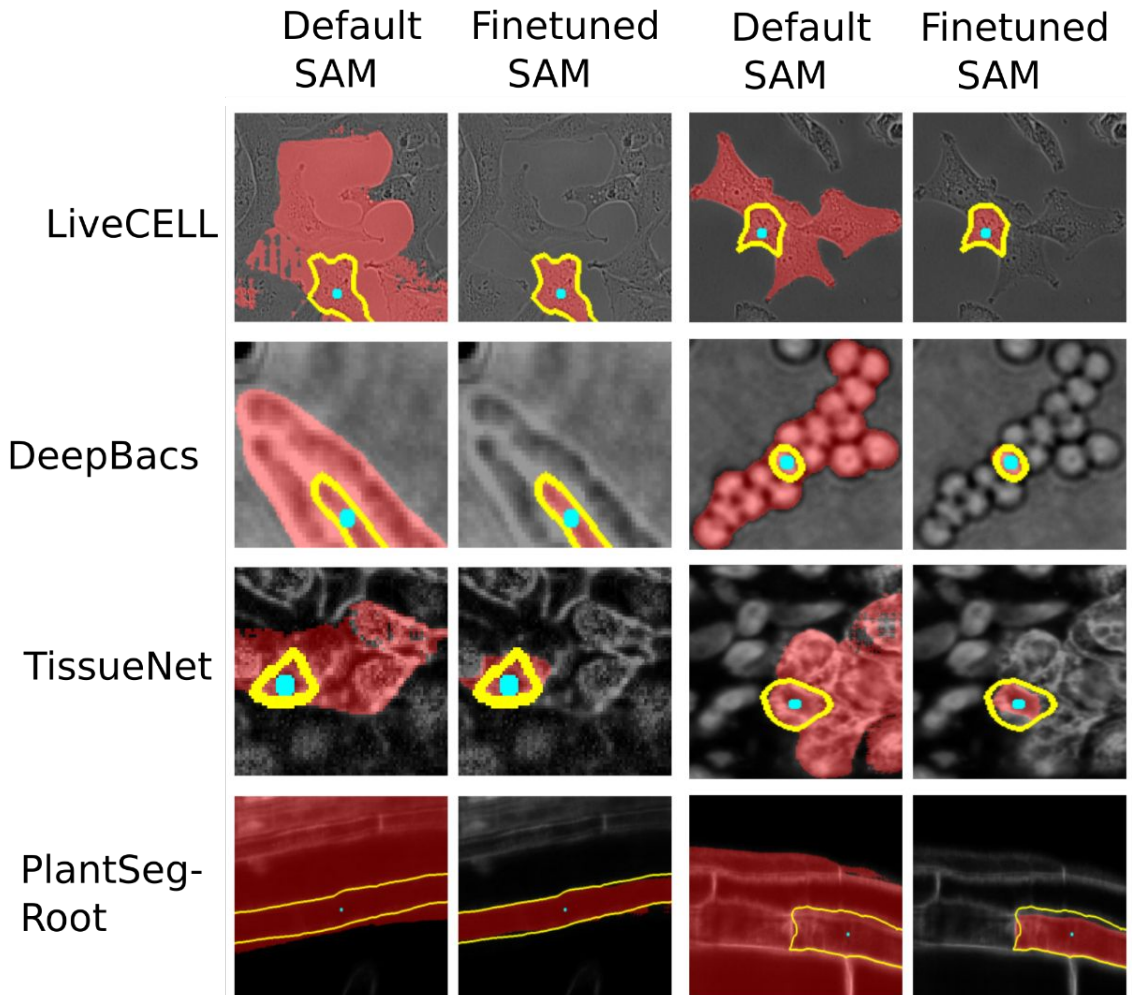
Finetuning for light microscopy

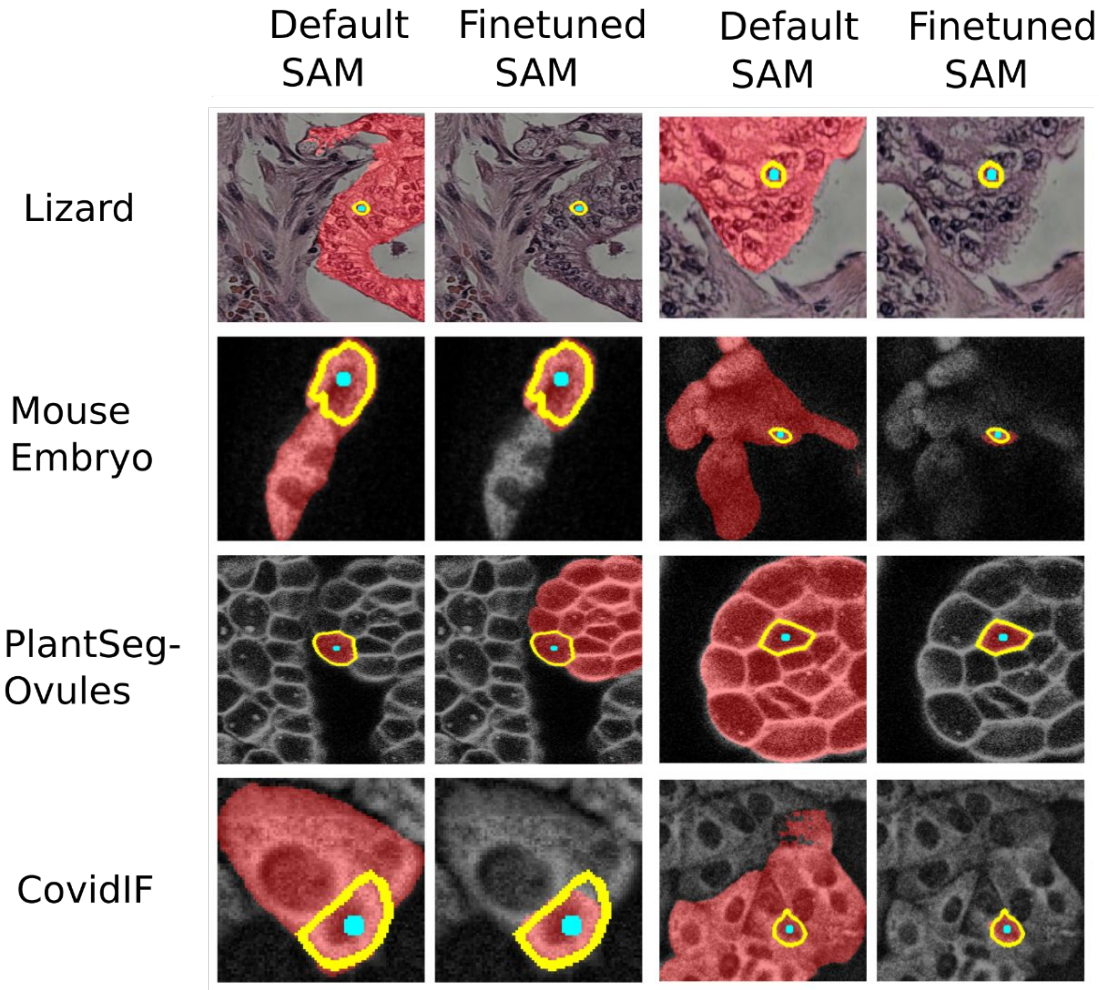
- Training data: cell and nucleus segmentation (published datasets)
  - Cells in Phase-contrast (LiveCELL)
  - Cells in Tissue (TissueNet)
  - Cells and Nuclei in Fluorescence (Neurips Cell Seg, DSB)
  - Cells in LightSheet (PlantSeg-Roots)
  - Bacteria in labelfree imaging (DeepBacs)

- Evaluate on test-split of training datasets (“in domain”)
  and unseen datasets (“out of domain”):
  - Nuclei and cells in confocal, cells in immunofluorescence, nuclei in histopathology, …

- Compare interactive and automatic instance segmentation
  - Compare to CellPose baseline for automatic segmentation

Interactive Segmentation: In domain & Out-of-domain





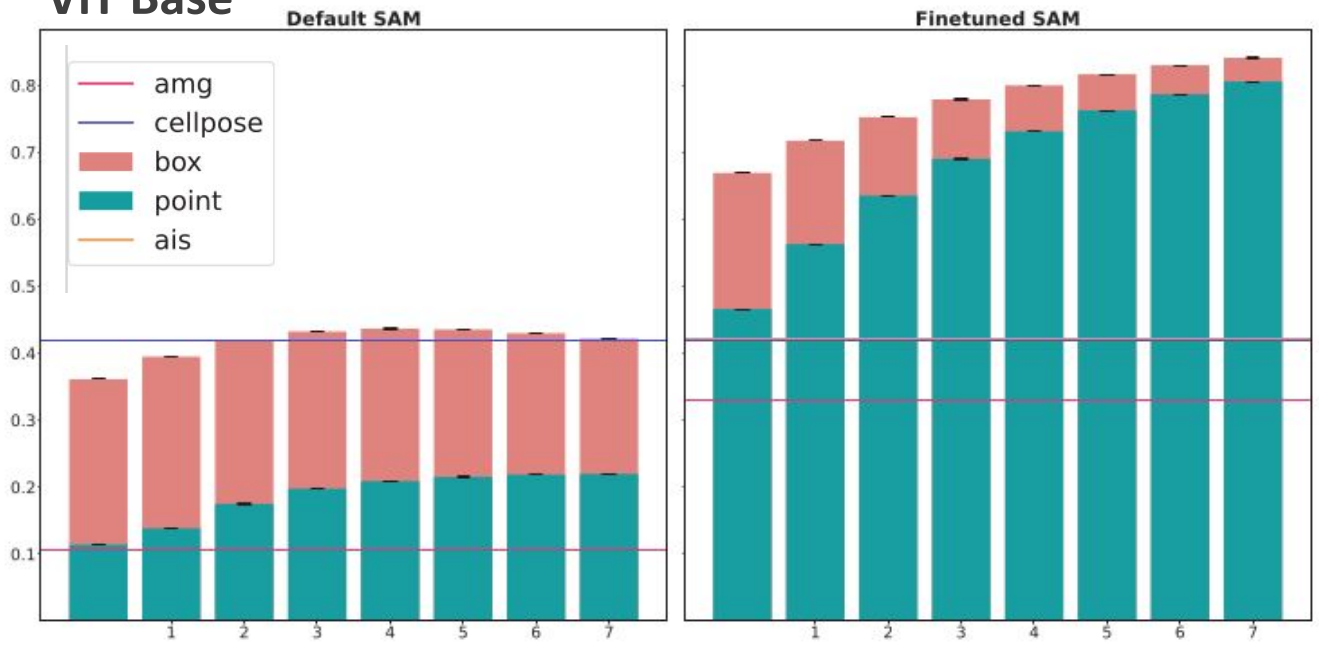
Results: In Domain

Results for LIVECell Dataset
(In Domain; Test Split)

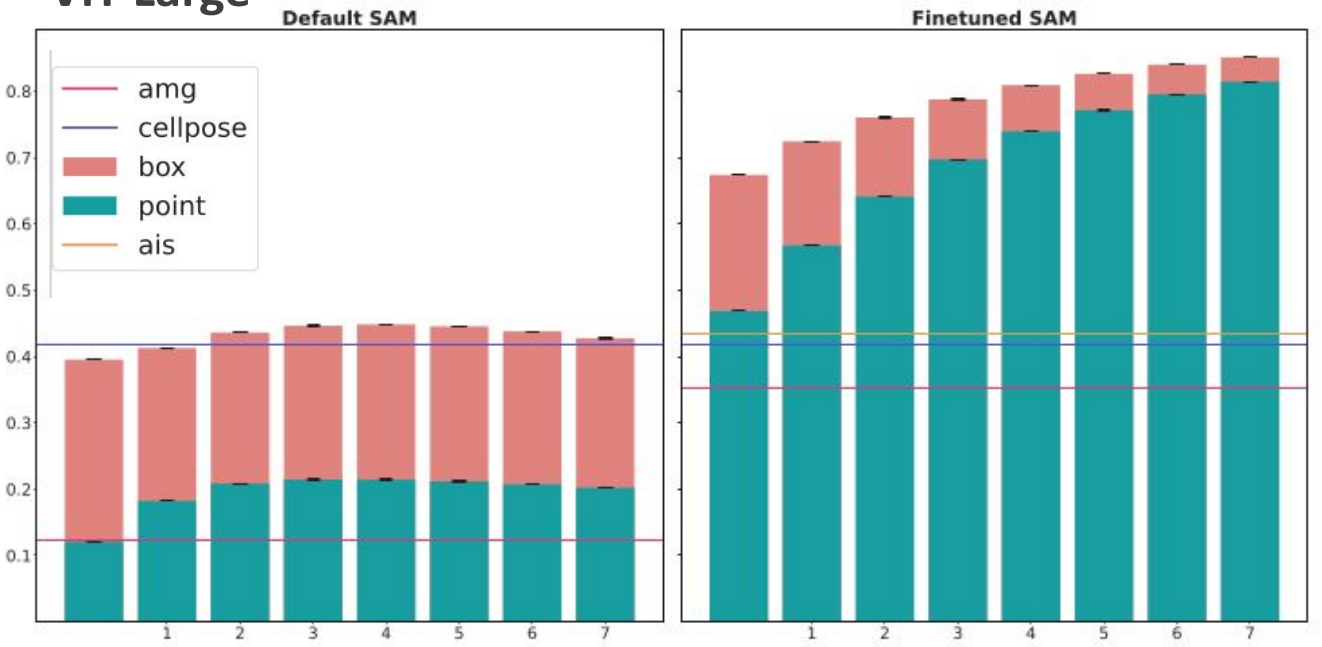
Evaluation:

- Interactive Segmentation:
  - Derive prompts from ground-truth, improve iteratively
- Instance segmentation:
  - Compare with CellPose
- Both: compute segmentation accuracy (compared to ground-truth)

ViT Base



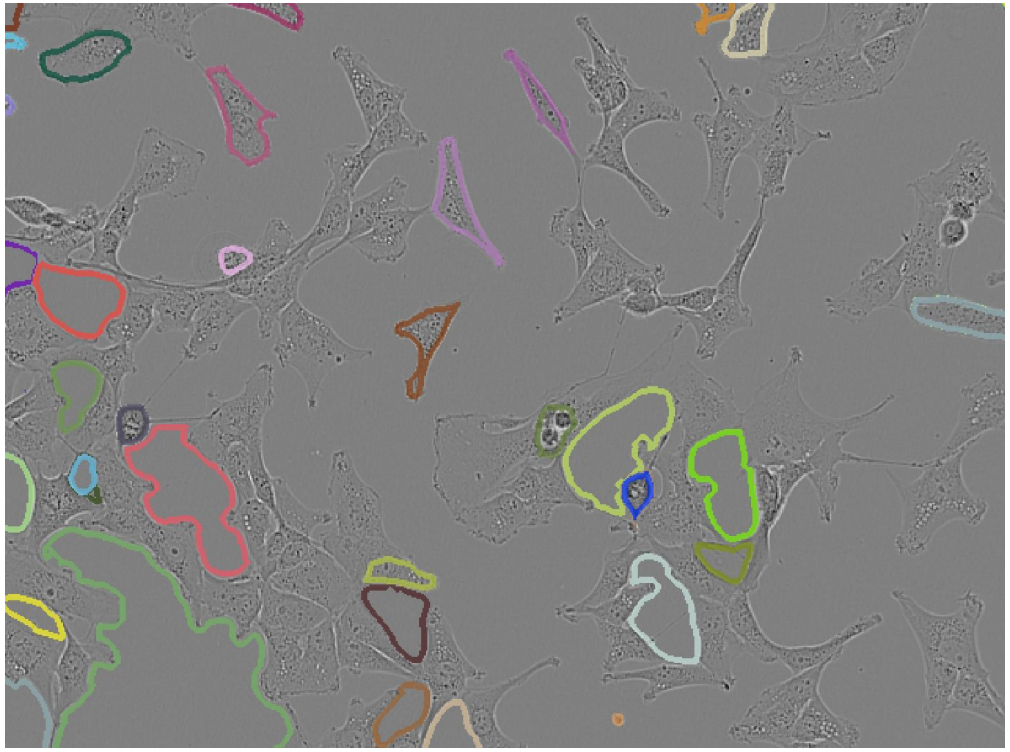
ViT Large



Automatic Segmentation

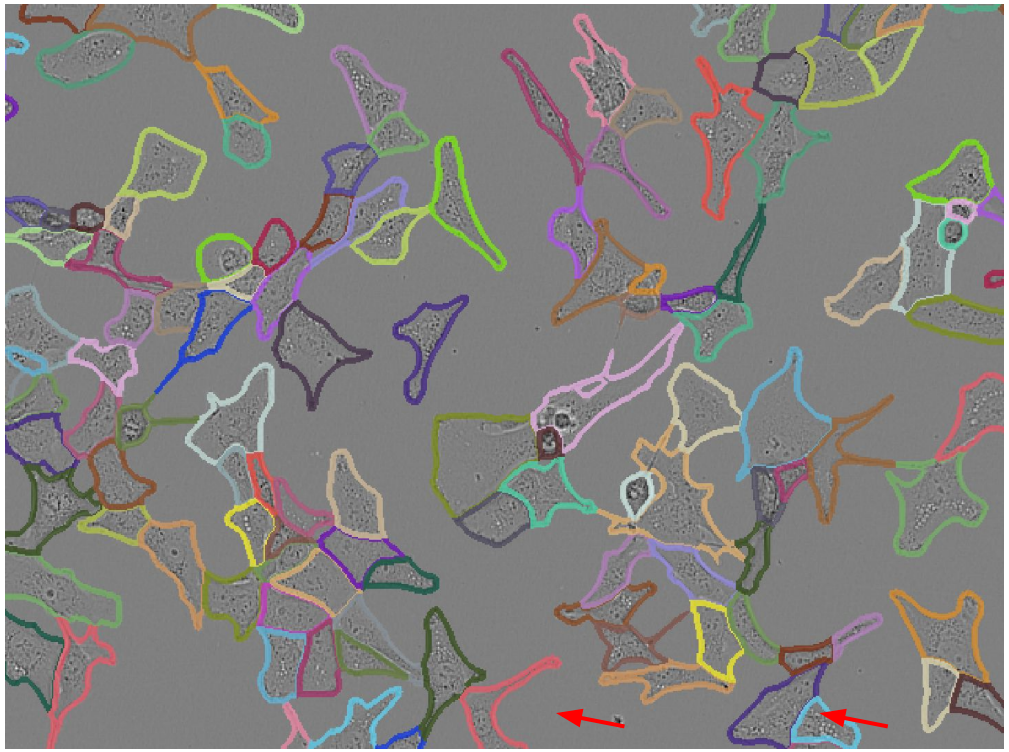
Instance segmentation on LIVECell Dataset

Runtimes on laptop (CPU);
including embedding computation (dominates for AIS)



VIT-B
AMG: 75 sec

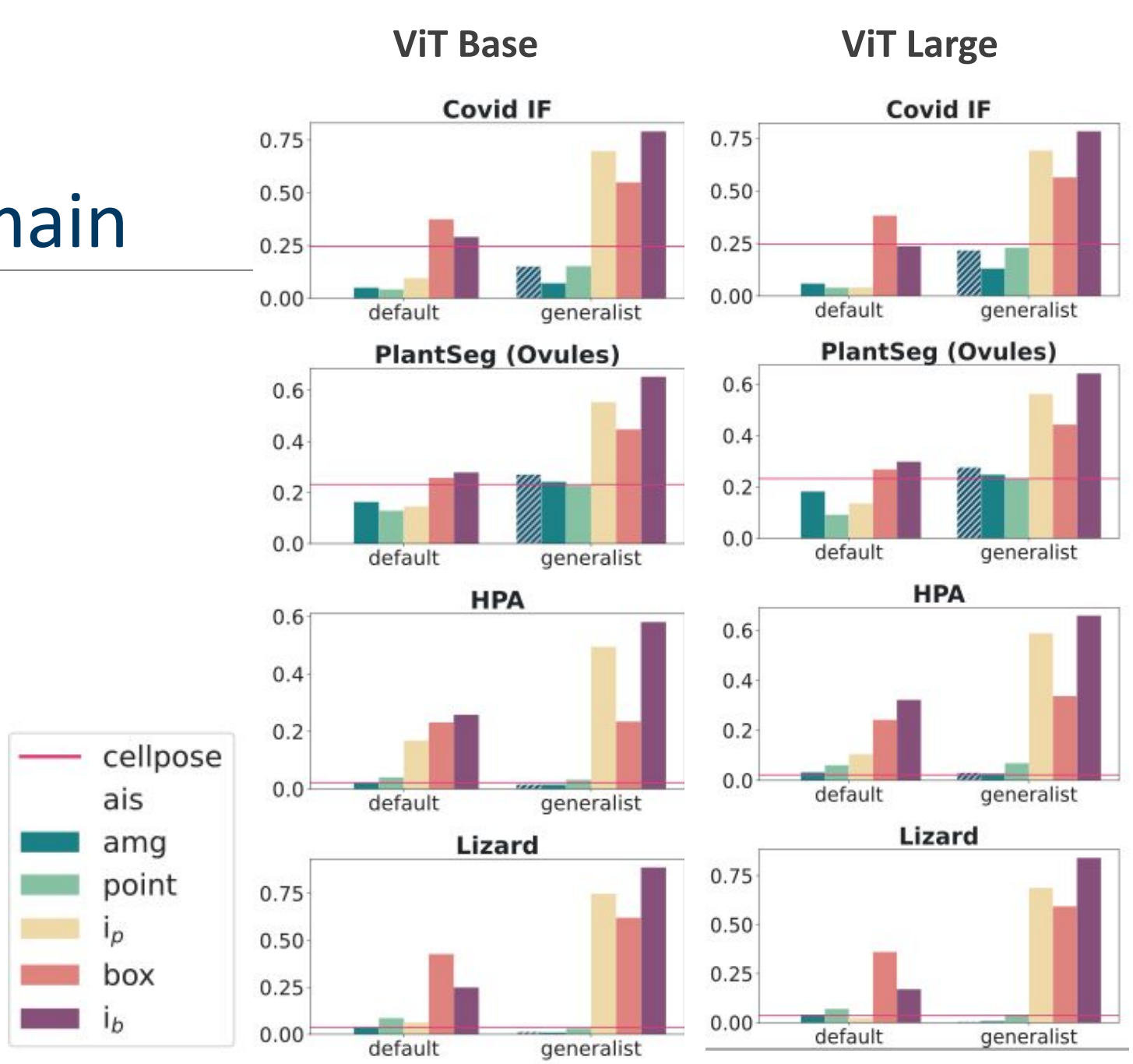
VIT-B-LM
AIS: 9 sec



Results: Out of domain

Results for out of-domain datasets.

Same evaluation procedure as before.



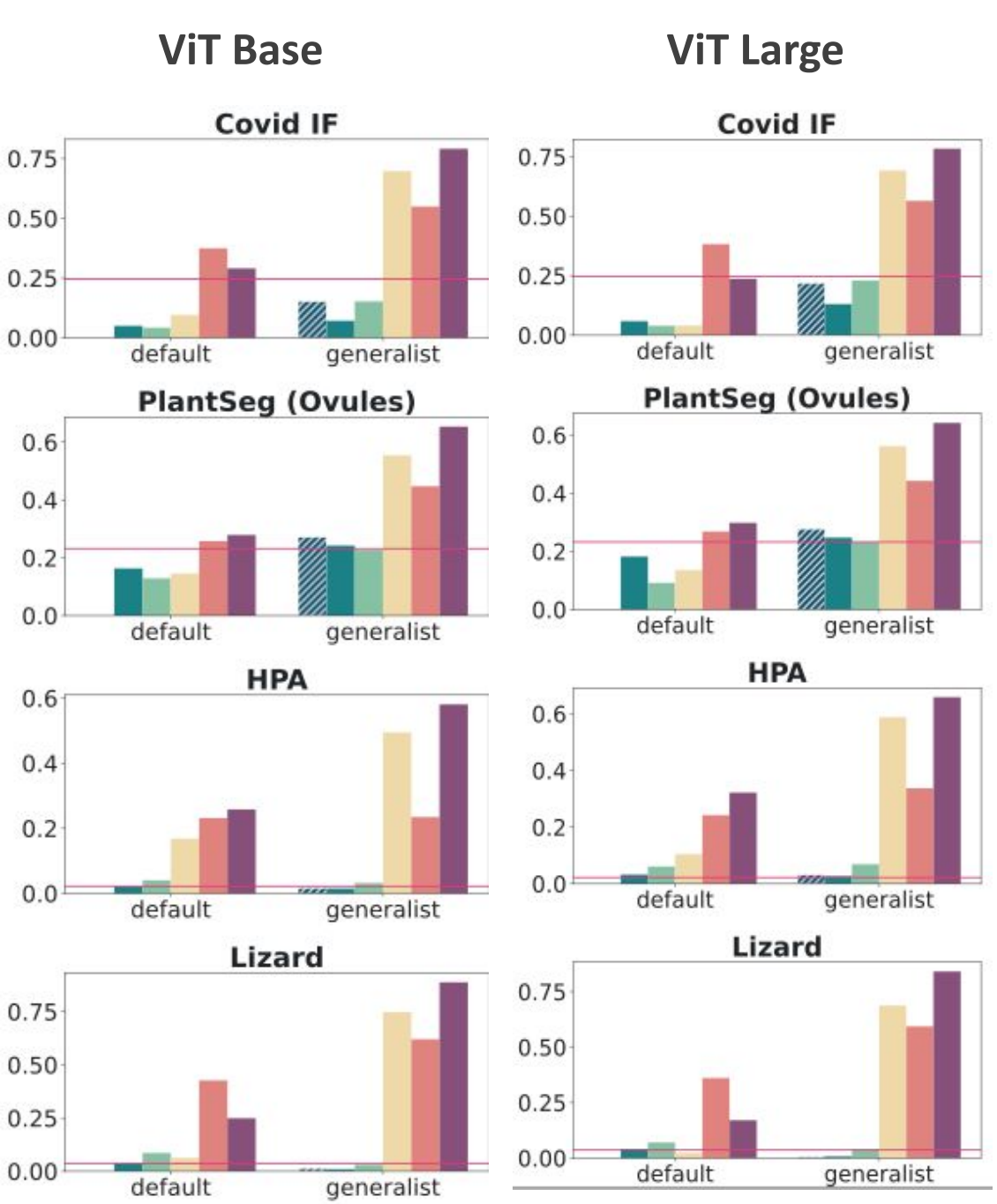
Results: Out of domain

Results for out of-domain datasets.

Same evaluation procedure as before.

Conclusions:

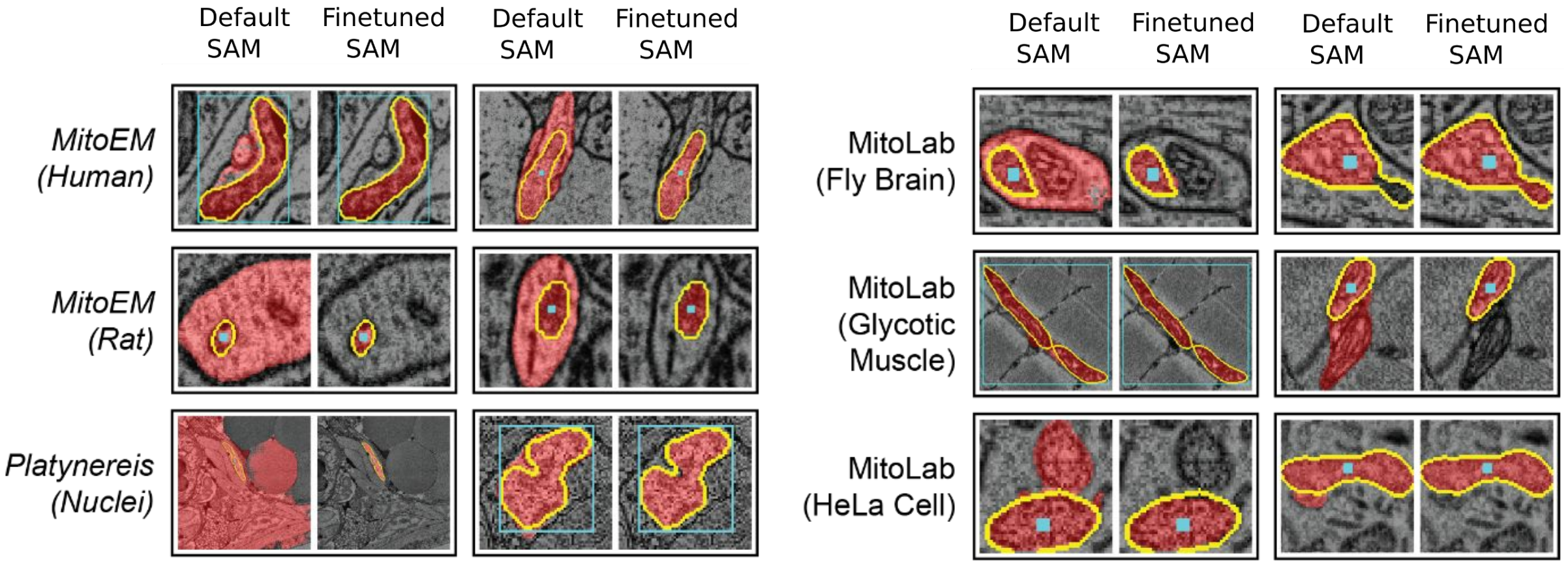
- Finetuning improves models!
- Best model: vit_l
  - If runtime matters: vit_b / vit_t
- Comparison to CellPose (automatic seg.):
  - Similar performance on most out of domain datasets (cyto2 model)



Finetuning for electron microscopy

- Training data: Mitochondria and nucleus segmentation in electron microscopy
  - Most training data from MitoNet (https://doi.org/10.1016/j.cels.2022.12.006).
- Compare default and finetuned model.
  - Compare automated segmentation with MitoNet.
- Evaluate on test-split of training datasets (“in domain”) and unseen datasets (“out of domain”)
  - Application to EM mitochondria from non-training data.

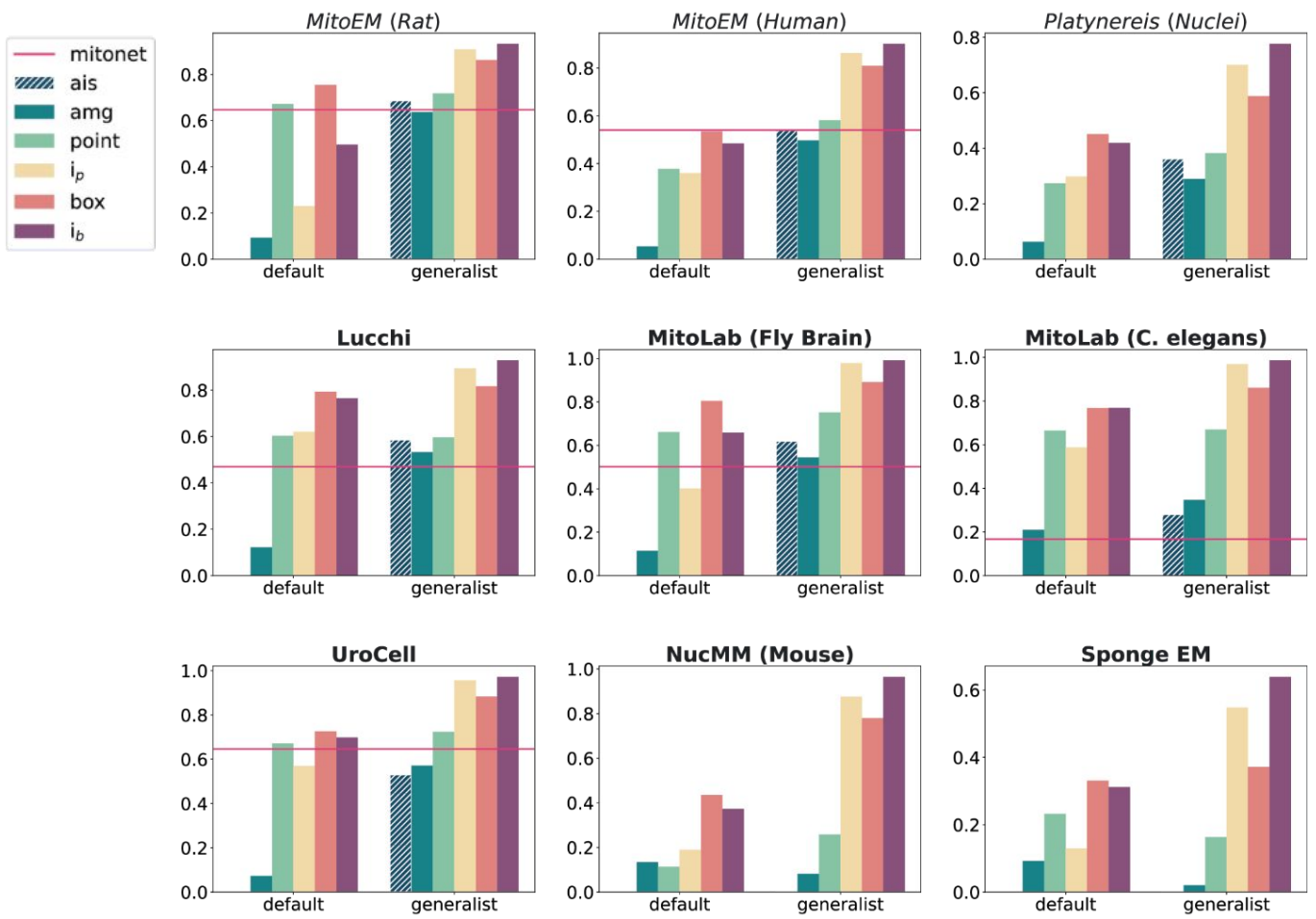
Interactive Segmentation: In domain & Out-of-domain



Results: In & out-of domain

Evaluation: Same approach as for LM

- In domain (top row)
- Out of domain (rest)



Results: In & out-of domain

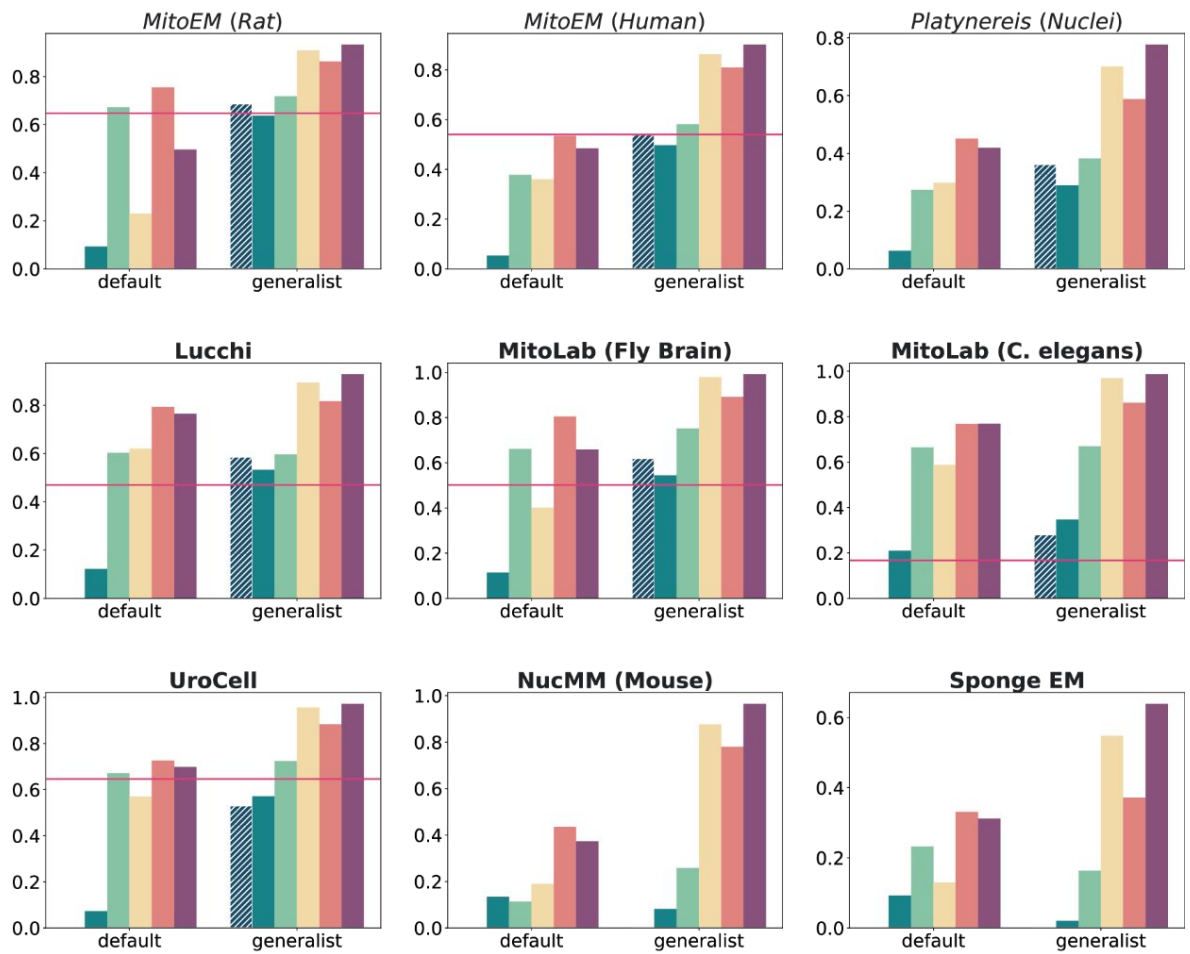
Evaluation: Same approach as for LM

- In domain (top row)
- Out of domain (rest)

Conclusions:

- Finetuning improves, best model is vit_l
- Similar performance to MitoNet on most datasets (AIS/AMG)

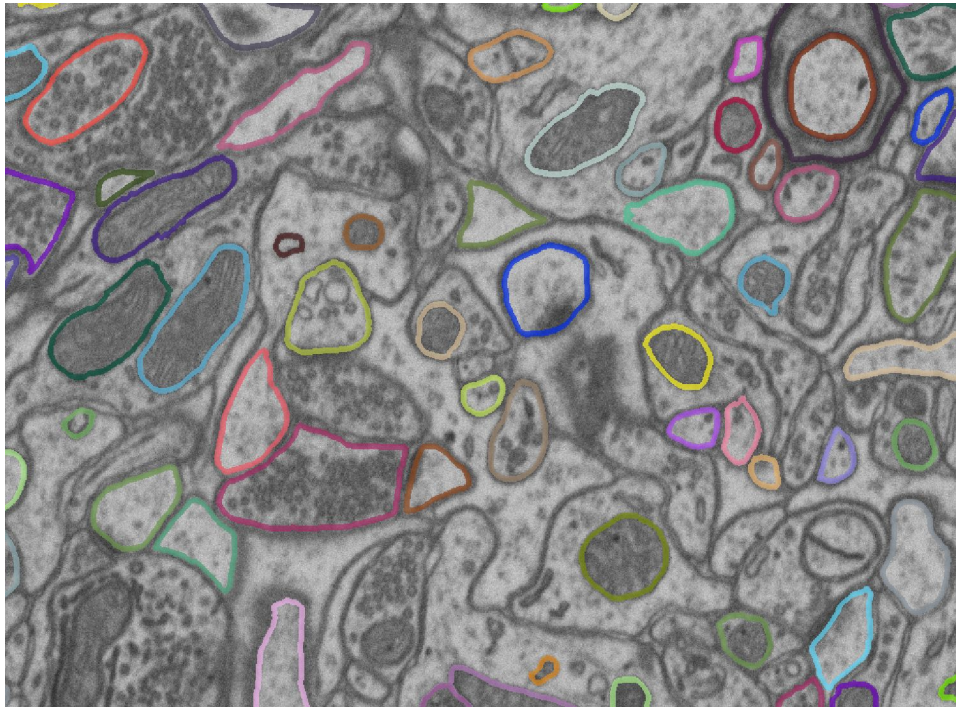
- **Improves segmentation for some other organelles (cilia, microvilli), but worsens it for cellular compartments**
  - **Bigger diversity in EM!**



Mitochondria

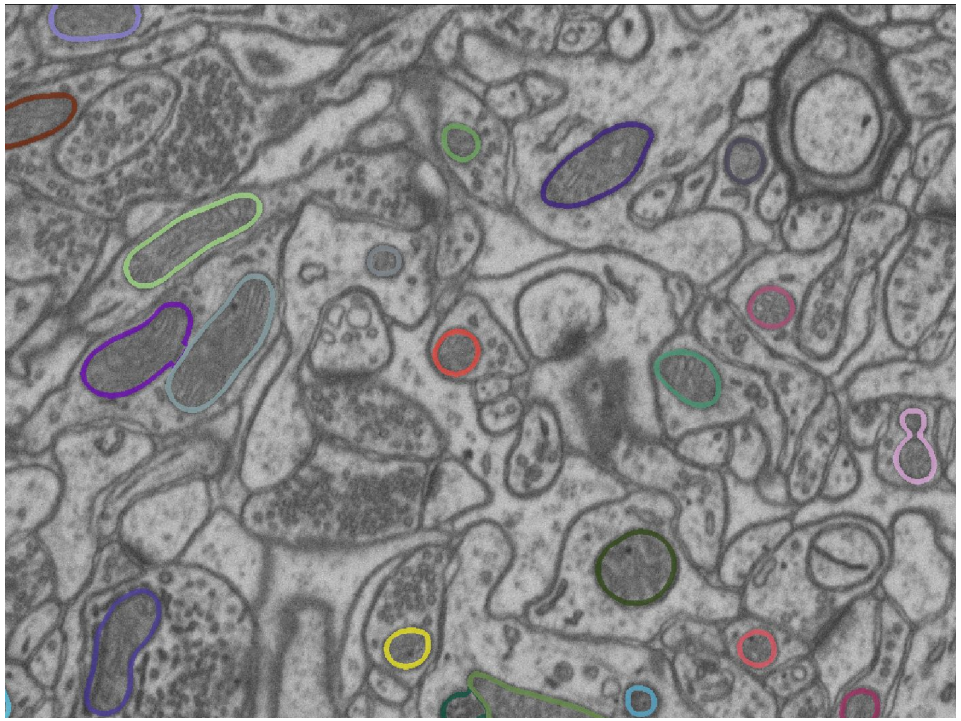
Instance segmentation on Lucchi Dataset

Runtimes on laptop (CPU);
including embedding computation (dominates for AIS)



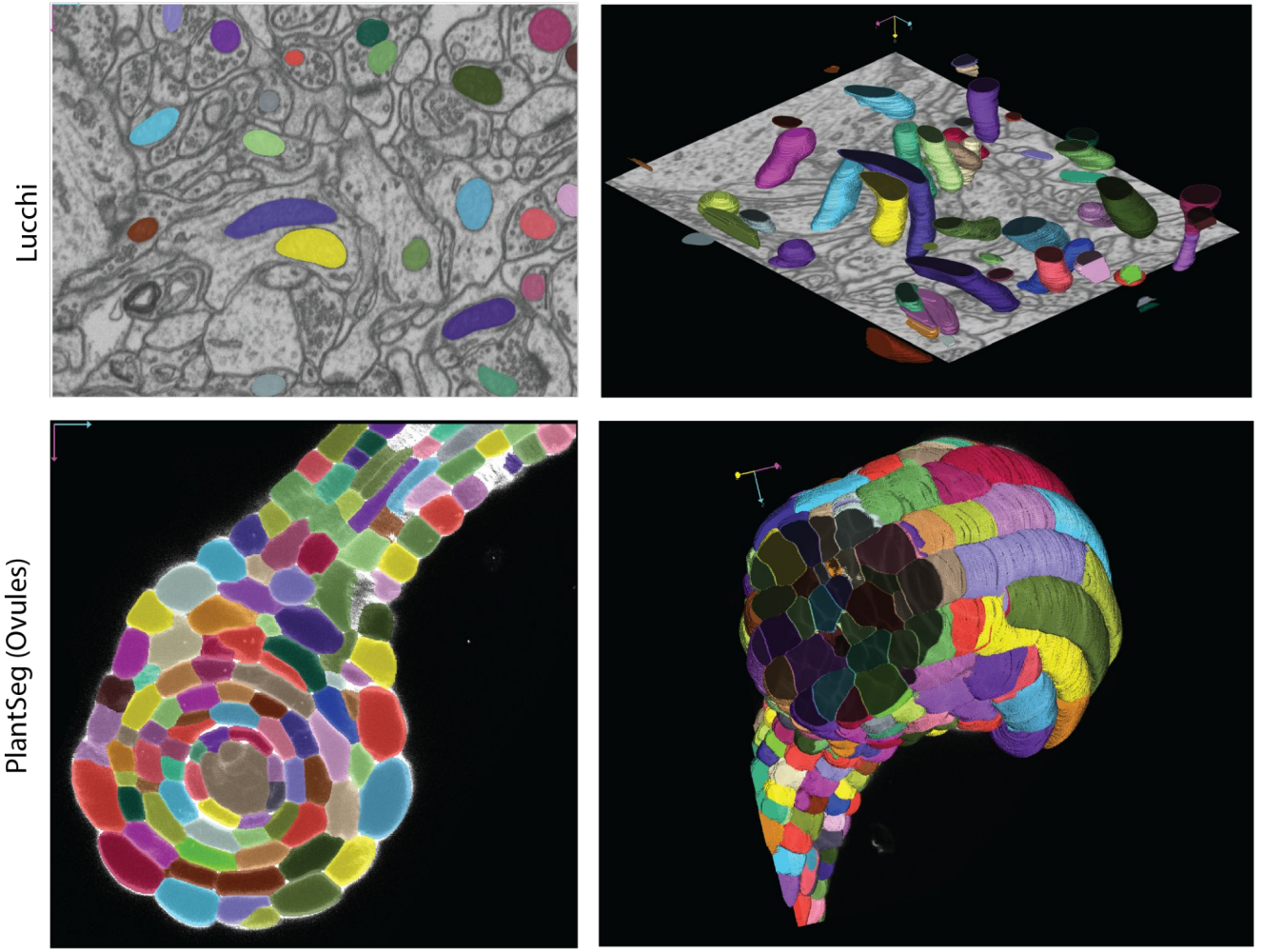
VIT-B
AMG: 80 sec

VIT-B-EM
AIS: 10 sec



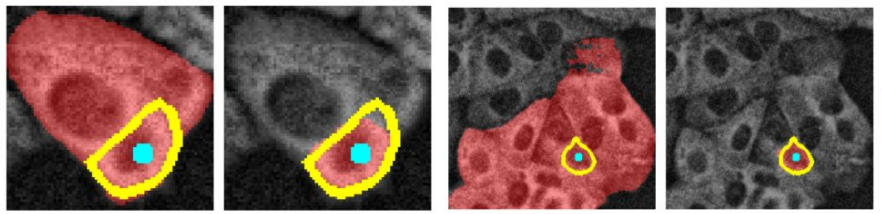
3D Segmentation with AIS

Segment objects slice by slice and merge across 3D



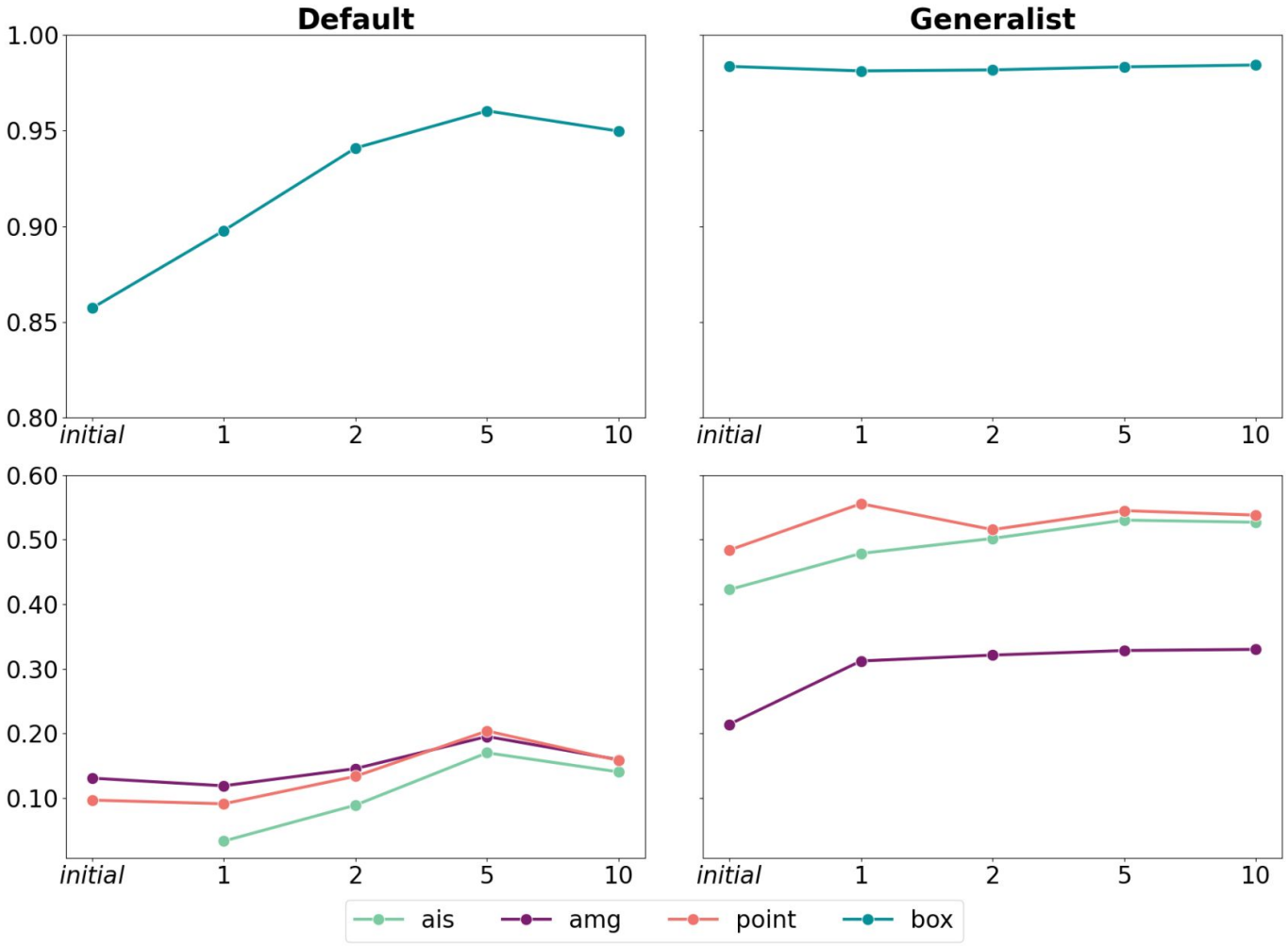
Finetuning as a user

CovidIF



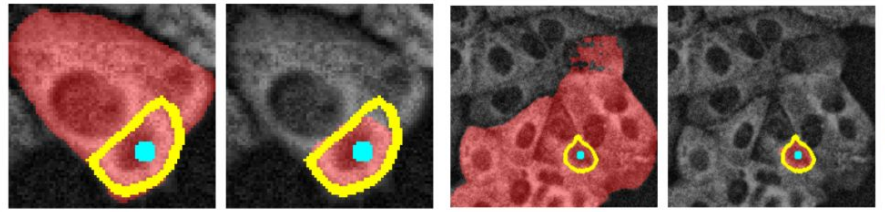
Improve models further for your data?

- How much data is needed?
- Which computational resources are required?



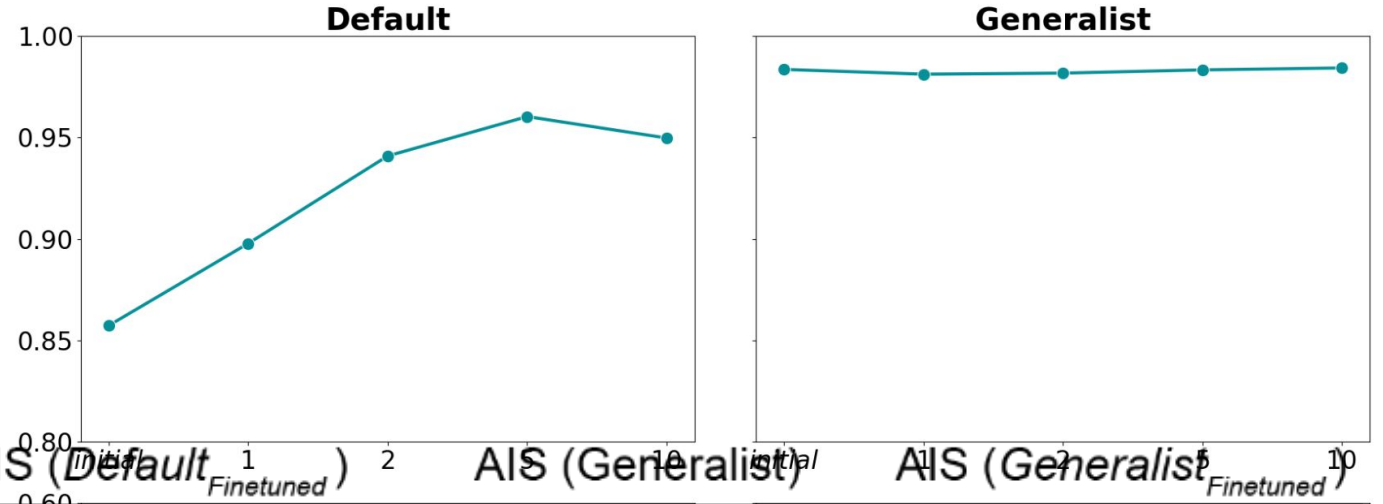
Finetuning as a user

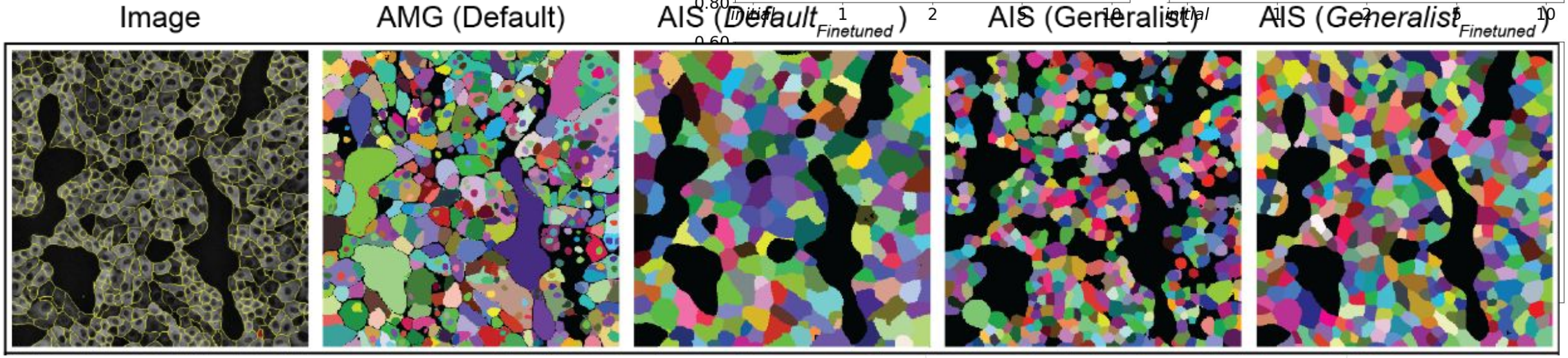
CovidIF



Improve models further for your data?

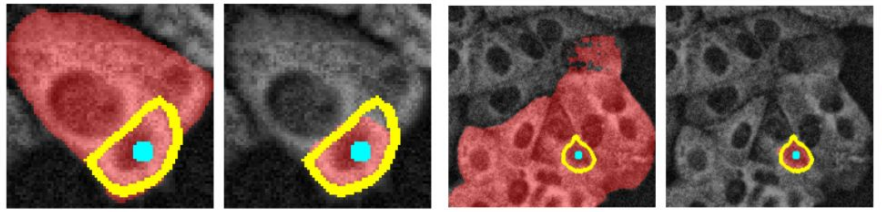
- How much data is needed?
- Which computational resources are required?





Finetuning as a user

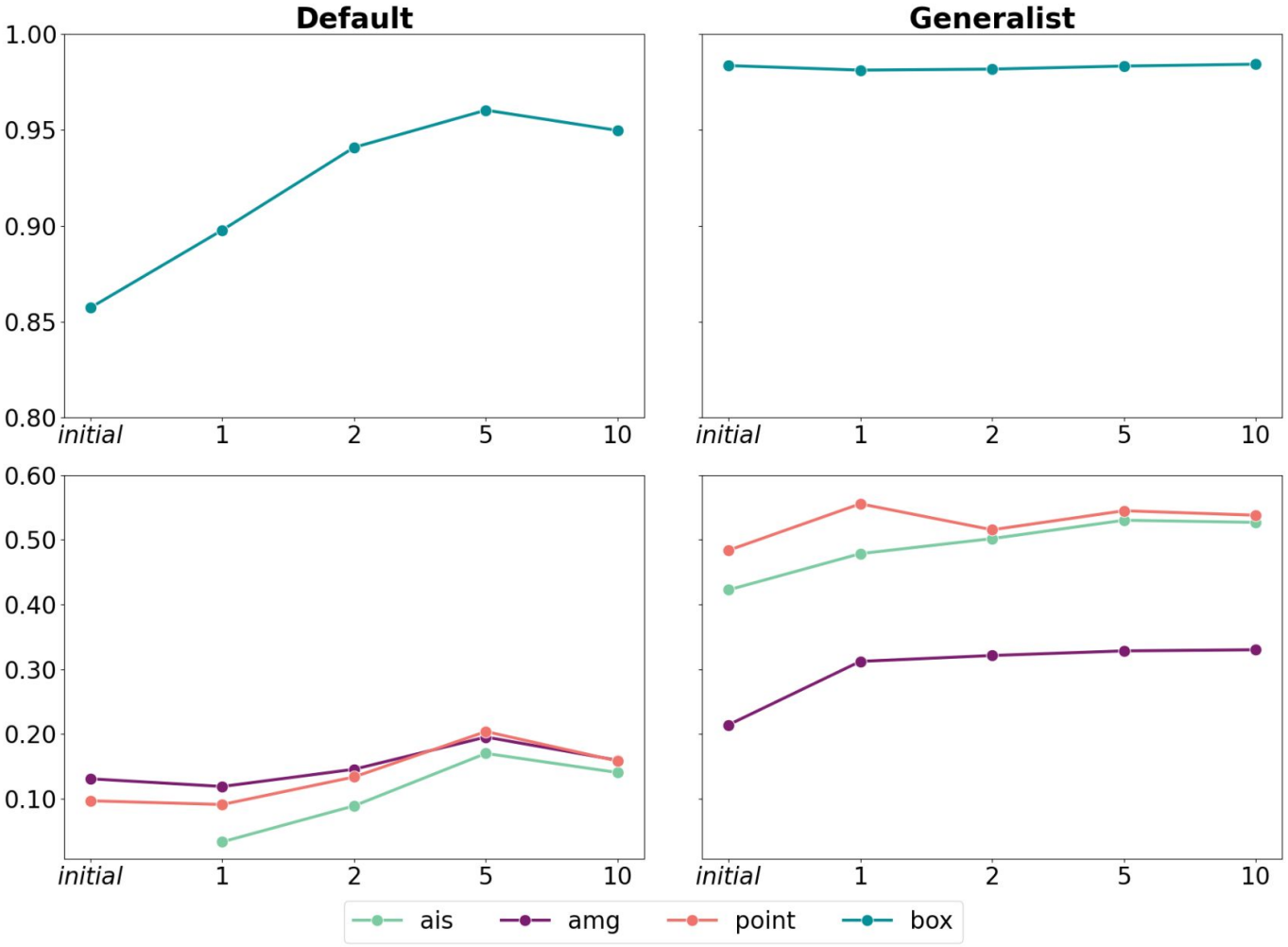
CovidIF



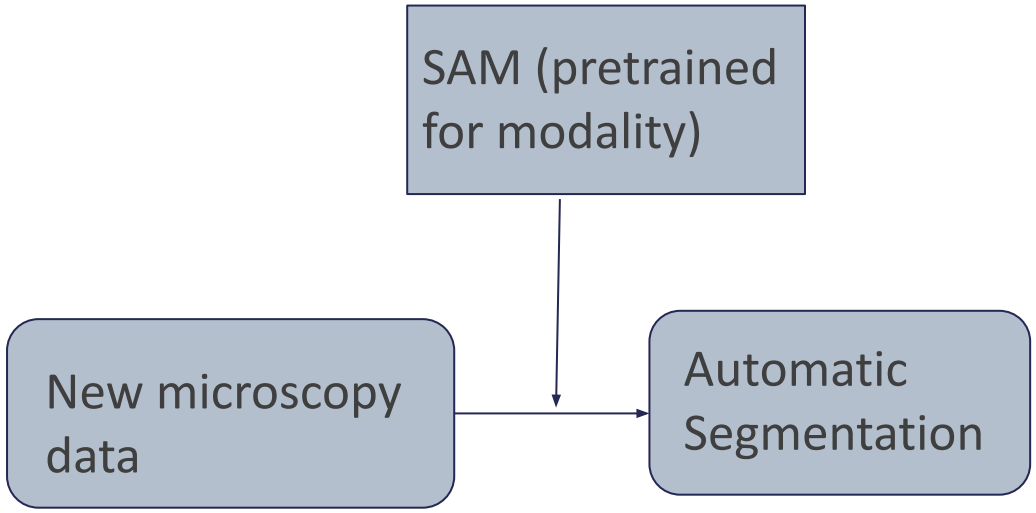
Improve models further for your data?

- How much data is needed?
- Which computational resources are required?

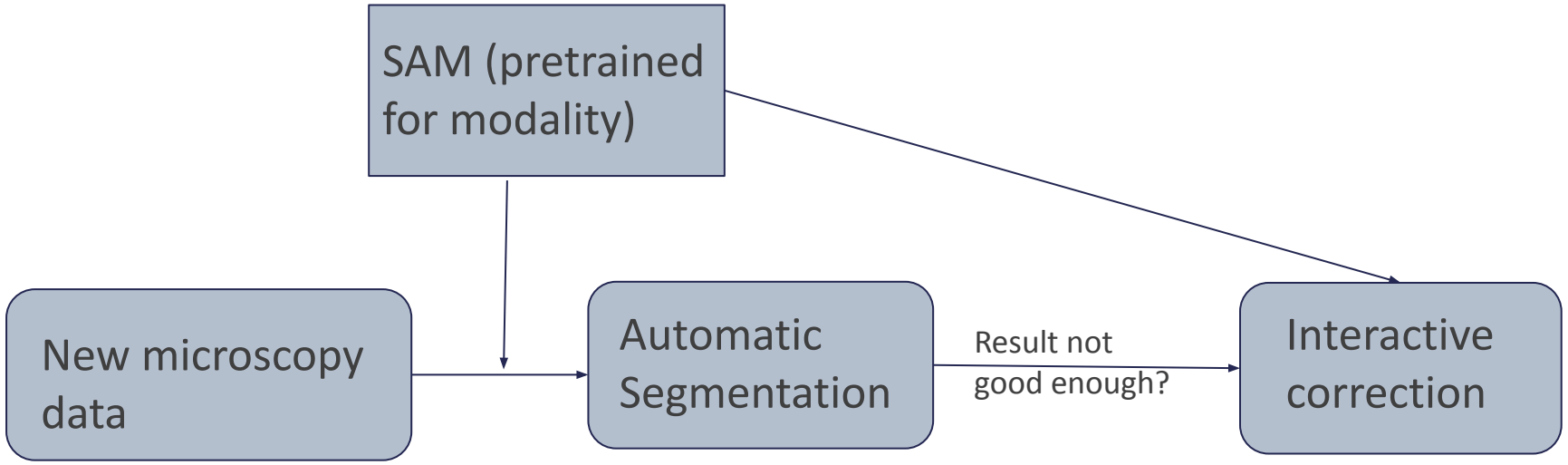
- Few images with annotations are sufficient!
- Possible on CPU
  - here: ca. 6 hours
- Faster on GPU
  - here: ca. 30 minutes



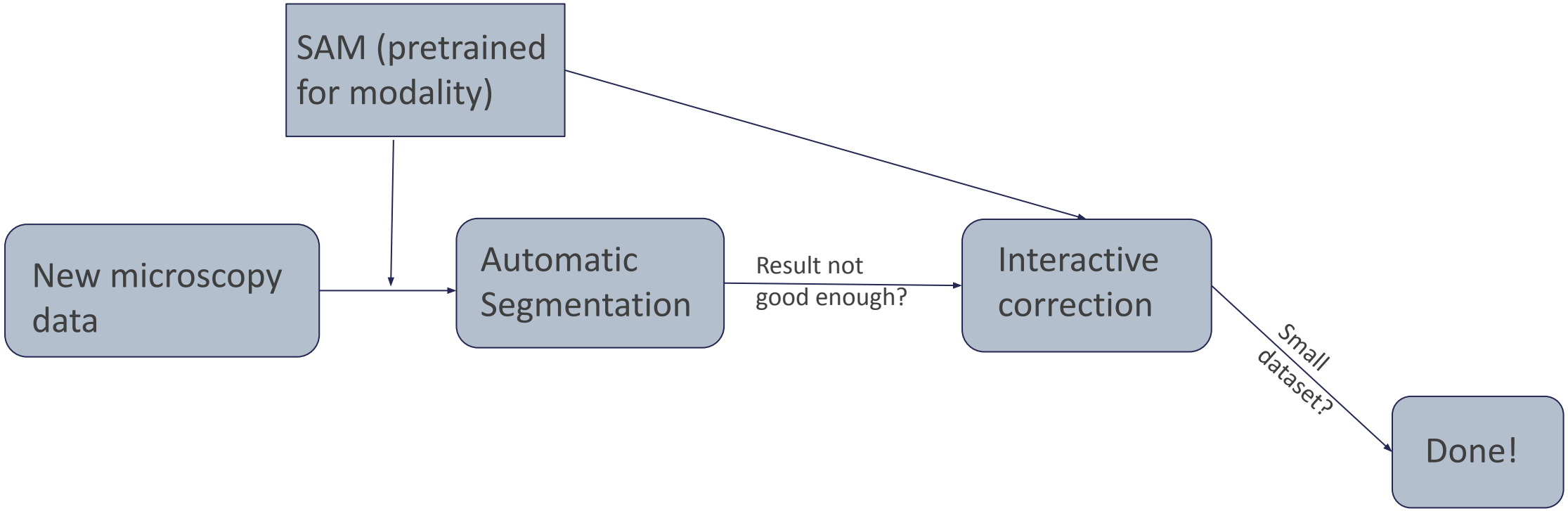
Application in practice



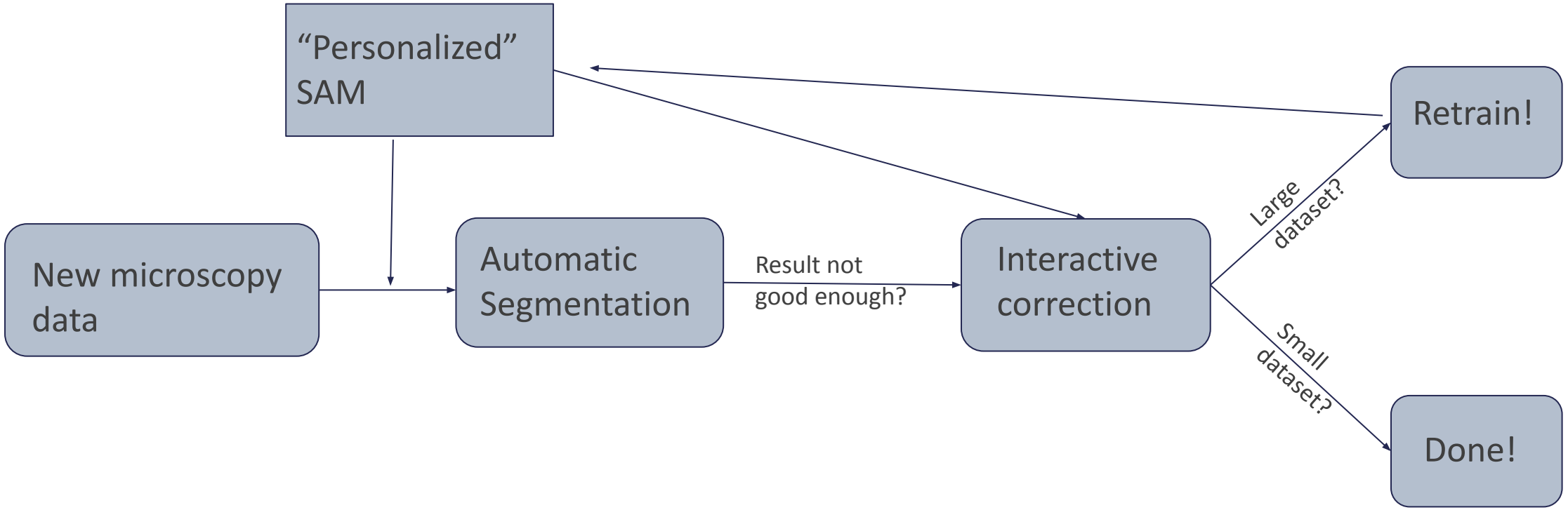
Application in practice



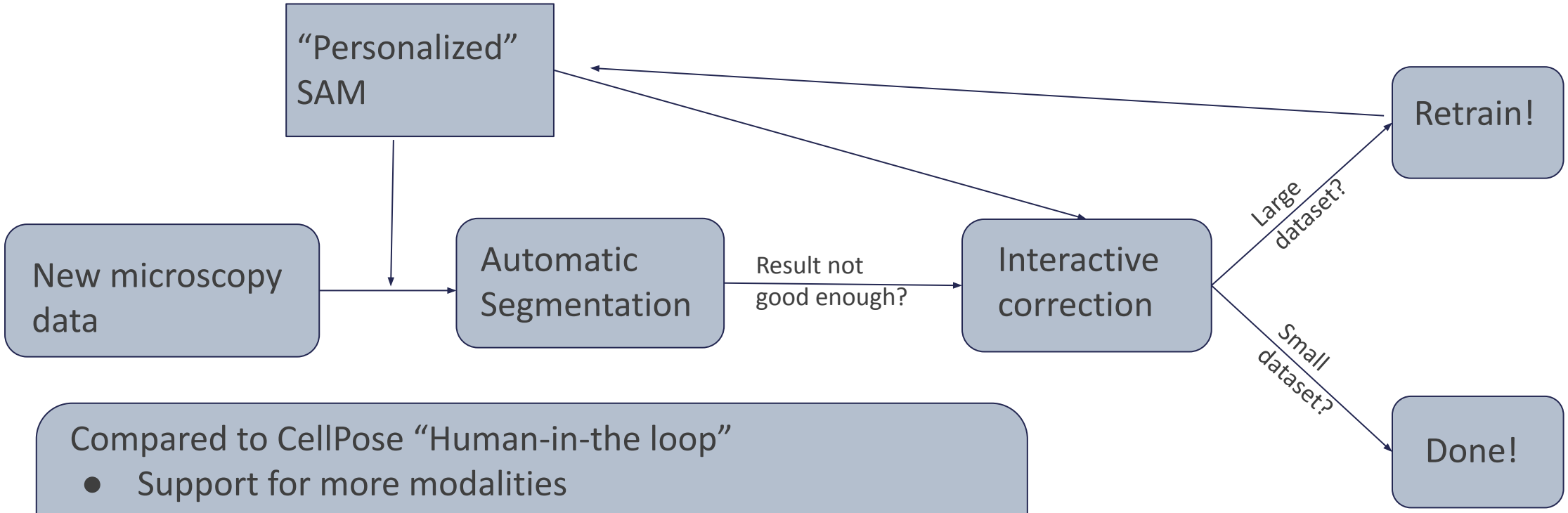
Application in practice



Application in practice



Application in practice



Compared to CellPose "Human-in-the loop"

- Support for more modalities
- Interactive correction speeds up annotation significantly!
- **BUT**: Training model takes longer (esp. on CPU)

Application in practice

Segmenting mitochondria in electron tomography with Wiebke Möbius and Leonie Schadt (MPI Göttingen)

Segmenting organelles in phytoplankton with Karel Mocaer (EMBL Heidelberg)

microSAM: Napari Integration

microSAM: SAM for napari

- napari plugins that enable interactive and automatic:
  - 2D Segmentation
  - 3D Segmentation
  - Tracking (2D + time)
  - Finetuning on own data

- Core functionality:
  - Default + generalist models
  - Multidimensional segmentation / tracking (interactive and automatic)
  - Tiled prediction for large images

microSAM: SAM for napari

- napari plugins that enable interactive and automatic:
  - 2D Segmentation
  - 3D Segmentation
  - Tracking (2D + time)
  - Finetuning on own data

- Core functionality:
  - Default + generalist models
  - Multidimensional segmentation / tracking (interactive and automatic)
  - Tiled prediction for large images

Code and documentation available at:
https://github.com/computational-cell-analytics/micro-sam

New release (v1.0):

- Latest microscopy models, compatible with BioImage.IO modelzoo.
- Updated and extended UI, napari plugin integration.

Parallel Developments & Next Steps & Outlook

https://arxiv.org/abs/2404.09556

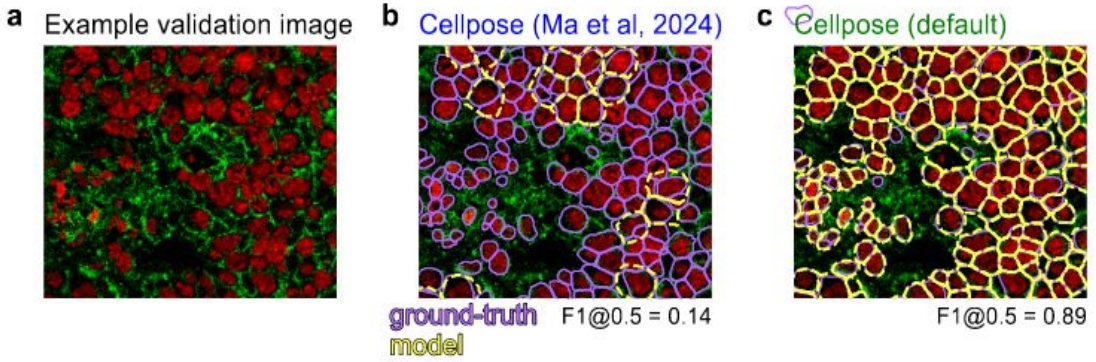
https://www.biorxiv.org/content/10.1101/2024.04.06.587952v1

Are vision transformers better than CNNs?

nnU-Net Revisited:
A Call for Rigorous Validation
in 3D Medical Image Segmentation

Transformers do not outperform Cellpose

Carsen Stringer[†], **Marius Pachitariu**[†]



https://arxiv.org/abs/2404.09556

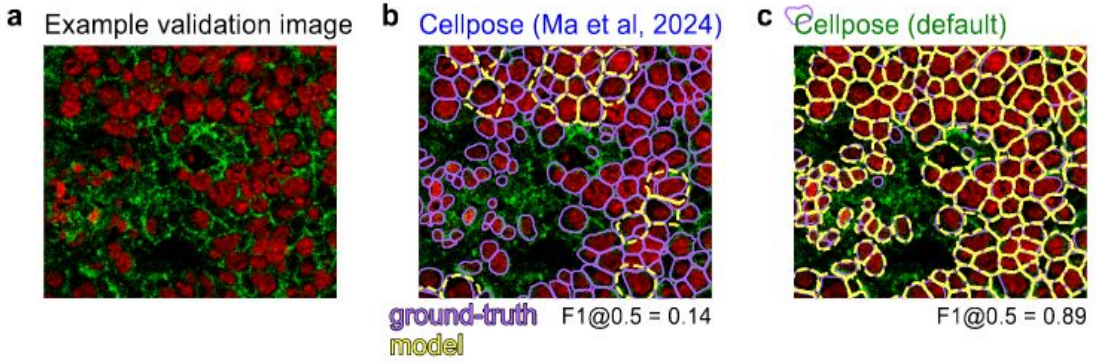
https://www.biorxiv.org/content/10.1101/2024.04.06.587952v1

Are vision transformers better than CNNs?

nnU-Net Revisited:
A Call for Rigorous Validation
in 3D Medical Image Segmentation

Transformers do not outperform Cellpose

Carsen Stringer†, Marius Pachitariu†



Transformers are not inherently better than CNNs!

- Disadvantages for small data and runtimes
- Possible advantage for large data
- Advantage of same architecture as NLP
  - Can profit from improvements and **easier to build multi-modal models.**

Feedback and contributions on the tool are very welcome!

Next steps

- Integration of efficient training procedures for finetuning (LoRA, **QLoRA**)
  - To enable better training on CPU and small GPUs
- Provide better and more models:
  - EM Organelle Generalist Model
    - Training on OpenOrganelle and other organelle segmentation datasets.
  - Histopathology Model

Check out our repository for all the details:
https://github.com/computational-cell-analytics/micro-sam

Outlook:
Universal microscopy segmentation and tracking

- Incorporate 3D (2D + time) segmentation in SAM-like model
  - Advantage ransformer: same model for 2d and 3d
- Vision Mamba: Investigate newer (more efficient) architectures
  - Our recent (**preliminary!**) work shows promise: https://arxiv.org/abs/2404.07705
- Semantic awareness (e.g. differentiate organelles in EM, one model for microscopy)
- Zero-shot adaptation (improve segmentation from examples)

Acknowledgments

EMBL Heidelberg
Anna Kreshuk & her group
et al.

Uni Göttingen & Campus
Alexander Ecker
Tobias Moser
Silvio Rizzoli
et al.

My group
Anwai Archit
Luca Freckmann
Sushmita Nair
Marei Freitag
Sagnik Gupta
et al.

Göttingen Campus

DFG

SFB 1286

Multiscale Cluster of Bioimaging Excellence

From Molecular Machines to Networks of Excitable Cells

CIDAS Campus-Institut Data Science

SARTORIUS

Hands-on Session

Before the break: Access VM, Install micro-sam

Access your VM and run:

$ mamba create -c pytorch -c nvidia -c conda-forge -n micro-sam pytorch micro_sam pytorch-cuda=11.6 -y

Plan Hands-On-Session

- Starting the tool, explain plugins
  - See https://youtu.be/gcv0fa84mCc for a rough idea
- 2D Segmentation on LiveCELL
  - Compare default and finetuned model (vit_b, show auto segmentation for vit_b_lm)
  - See https://youtu.be/9xjJBg_Bfuc for a rough idea
- 3D Segmentation on Lucchi
  - Segment mitos in 3d with the em model
  - See https://youtu.be/nqpyNQSyu74 for a rough idea
- If there is still time students can choose to:
  - Try annotation on their own data
  - Start finetuning a model, either on their own data or on our sample data. Using https://github.com/computational-cell-analytics/micro-sam/blob/master/notebooks/sam_finetuning.ipynb

Application on your data

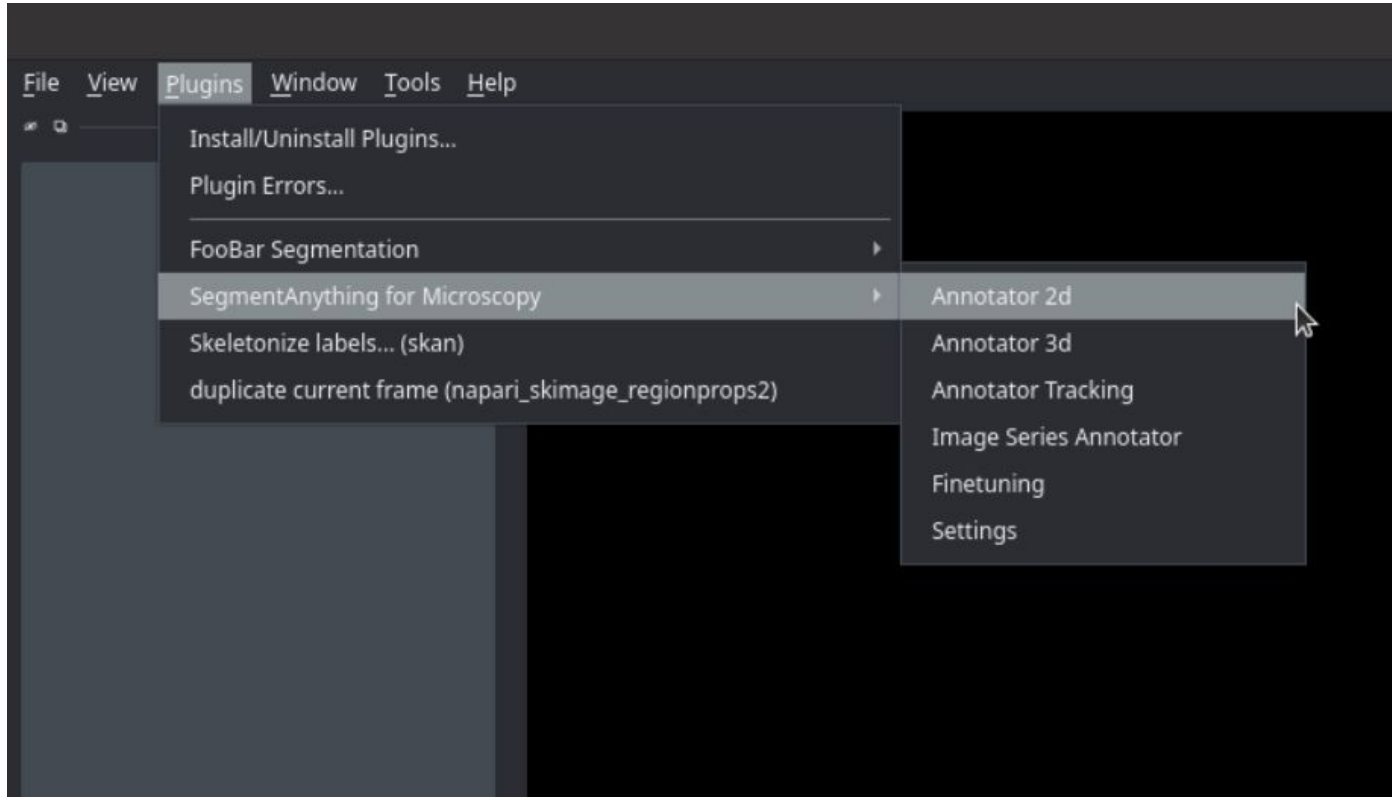
- 2D segmentation -> “Annotator 2d” or “Image Series Annotator”
- 3D segmentation -> “Annotator 3d”
- Tracking -> “Annotator Tracking”

Which model should I use?

LM -> vit_X_lm

EM (potato-shaped structures)
-> vit_X_em_organelles

Other -> vit_X



Finetuning Notebook

https://github.com/dl4mia/03_learned_representations/blob/main/sam_finetuning_HT.ipynb

To get it on your VM:

$ git pull origin main