Layer Query Networks
For
Test Time Training

many thanks to Jason for help :-)

rajat

Where I am in the process

- 3/3/2/2 NeurIPS, Rating scale 6.
- Reviewers liked idea, concerns on clarity
  - My writing was not clear, could be better.
  - Too vague, and ambitious in some places.
- Withdrew
- Resubmitting toned down version to AAAI Phase 1-> ICLR.
- **Any Feedback greatly welcome :-)**

- Classically:
  - Neural nets are trained on a dataset.
  - Weights are kept fixed during deployment.
- Test Time Training (TTT)
  - Train the network even during testing.
  - Can we learn on a single sample?

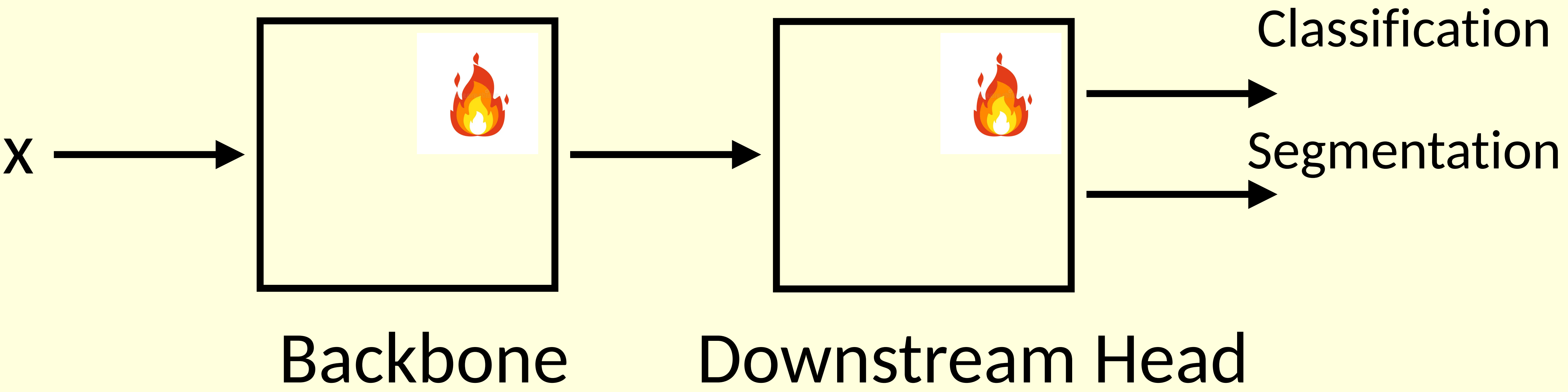
- First, i will describe what is TTT
- And then go into problem statement

Clarifying Experimental Setup

- A brief overlook at **TEST TIME TRAINING** setup.
  - Originally, introduced by Yu Sun & Alexei Efros (Berkeley)

https://yueatsprograms.github.io/ttt/home.html

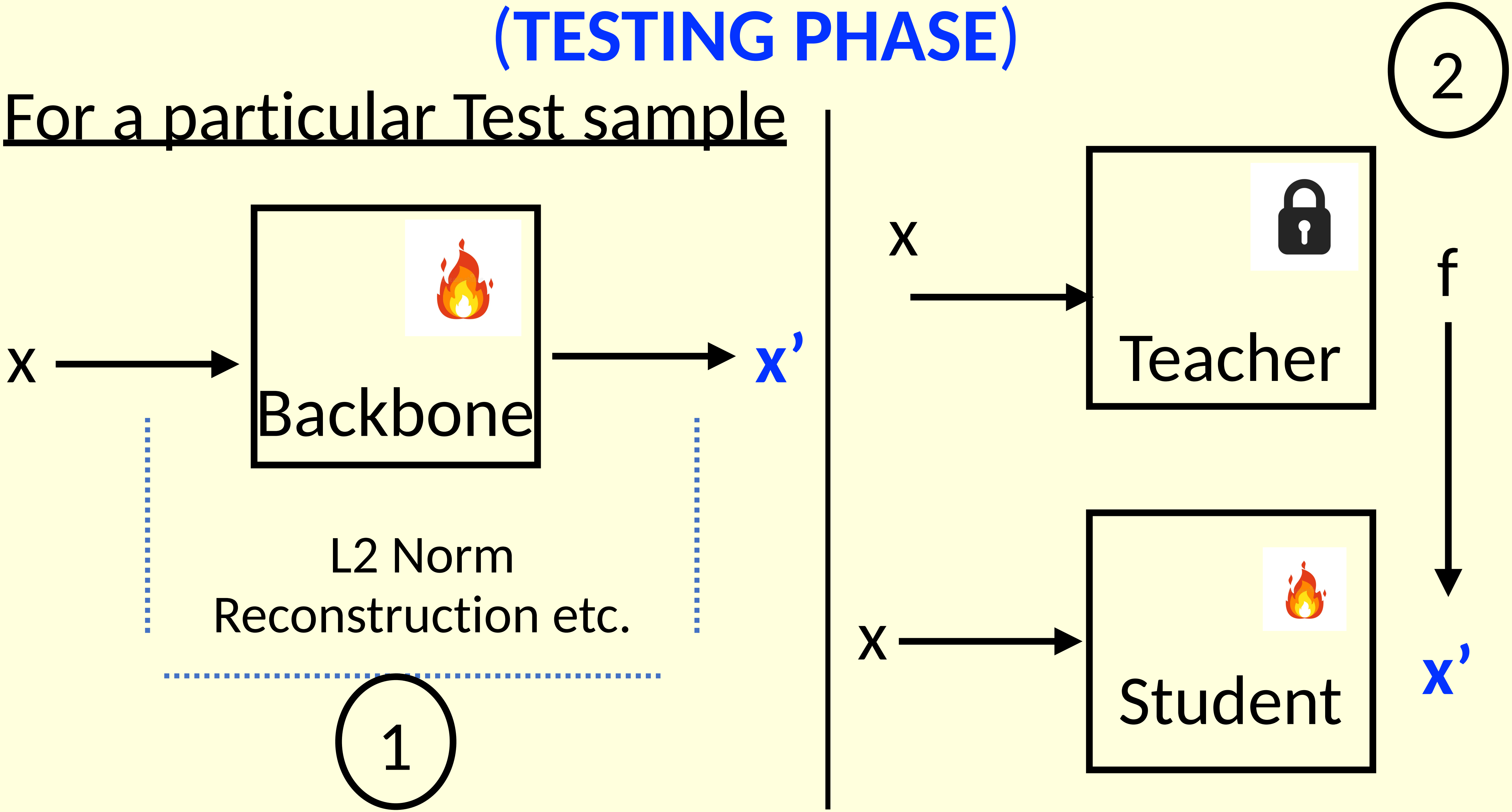
TEST-TIME-TRAINING SETUP
(**TRAINING PHASE**)



- Done on Train set.

TEST-TIME-TRAINING SETUP
(TESTING PHASE)

For a particular Test sample

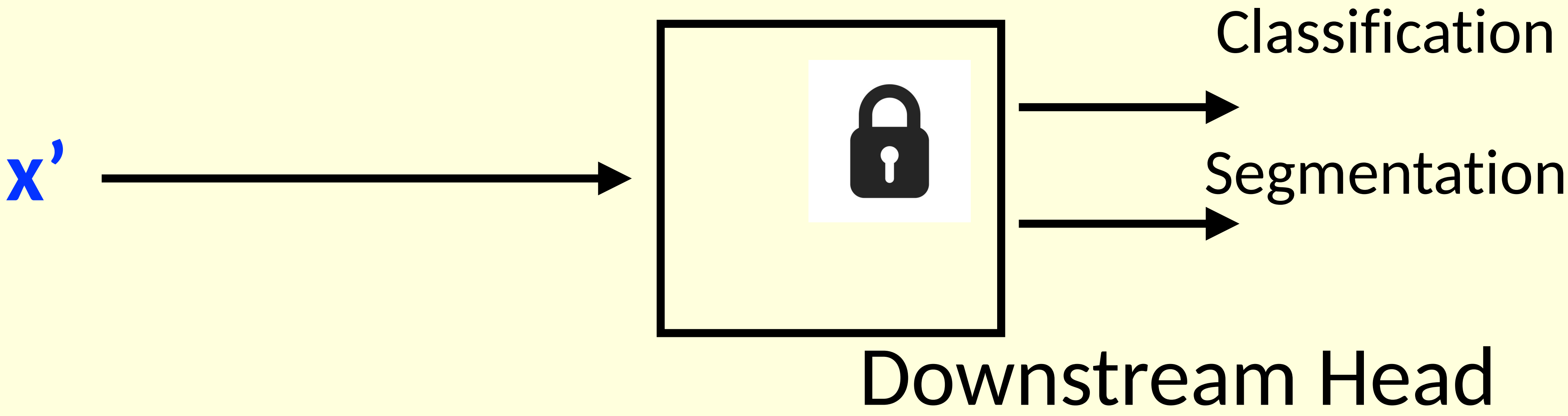


TEST-TIME-TRAINING SETUP
(**TESTING PHASE**)

These steps are repeated for many iterations

- In practice, 15-20 iterations.

TEST-TIME-TRAINING SETUP
(**TESTING PHASE**)

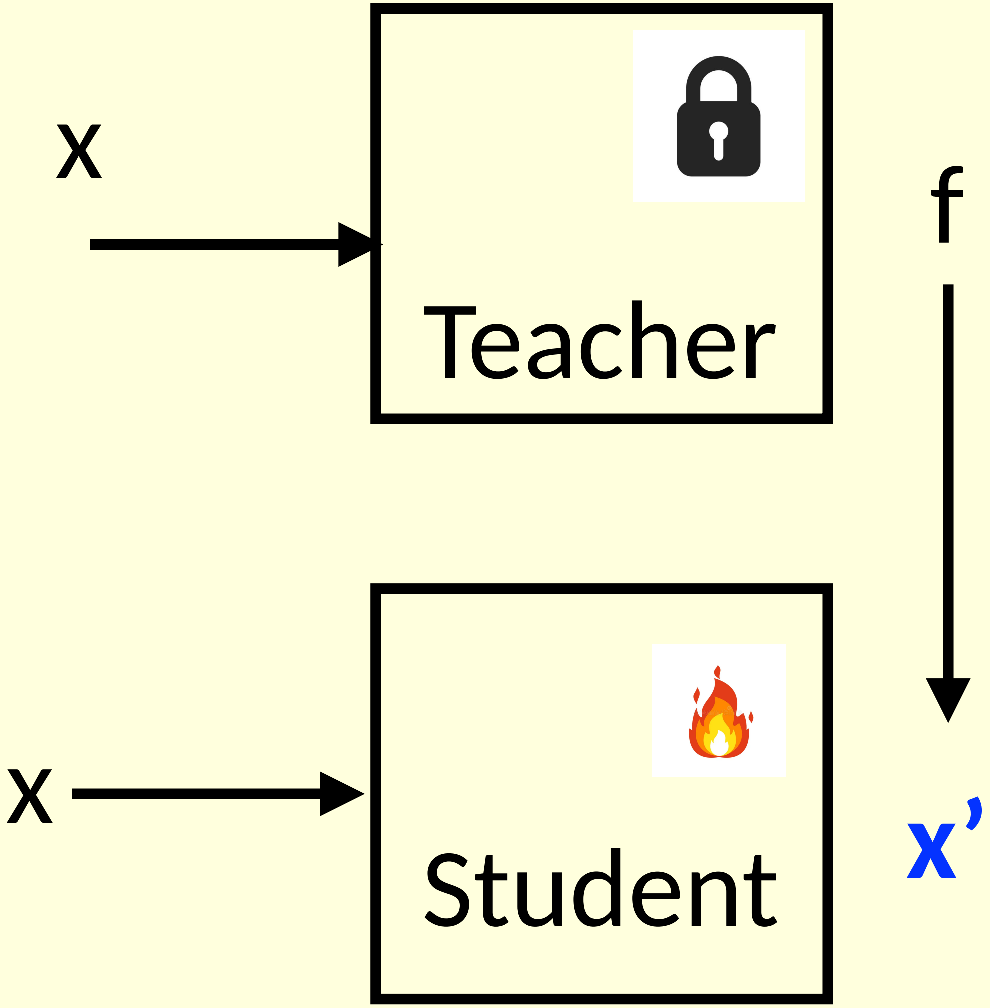


- The Downstream Head never sees Test-labels.
- x' is inference without any test labels.
- The claim is that x' (features with TTT) are better than x. So better downstream performance.

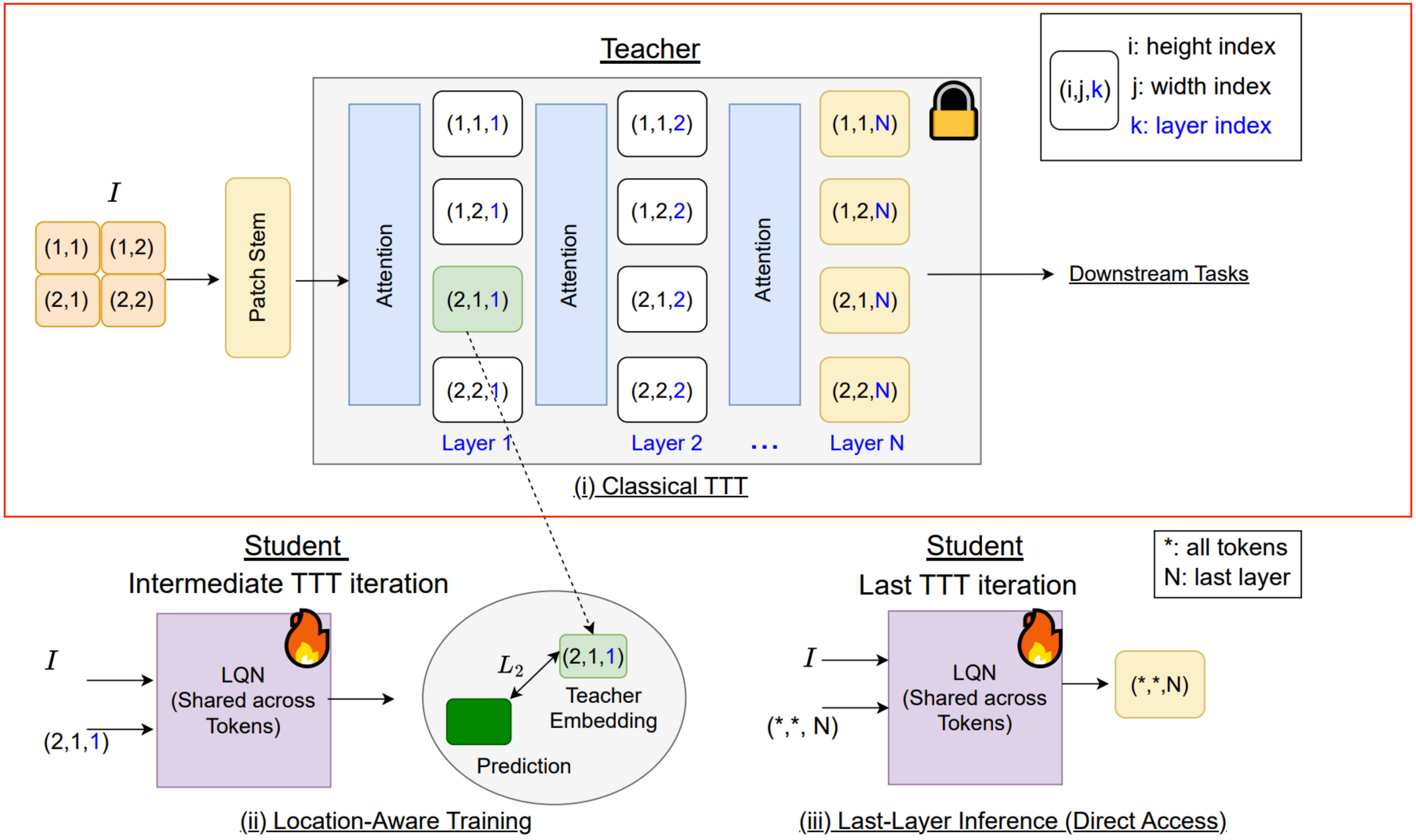
PROBLEM STATEMENT

- Suppose you want to distill all features of teacher. All the layers.
- M1: Student architecture = Teacher. (Fwd pass slow, since teacher = VIT)
- M2: Distill only last layer. (Cant use intermediate feature representations) **Key q: How to distill all layers without running into computational bottlenecks.**
- Chicken and egg problem.

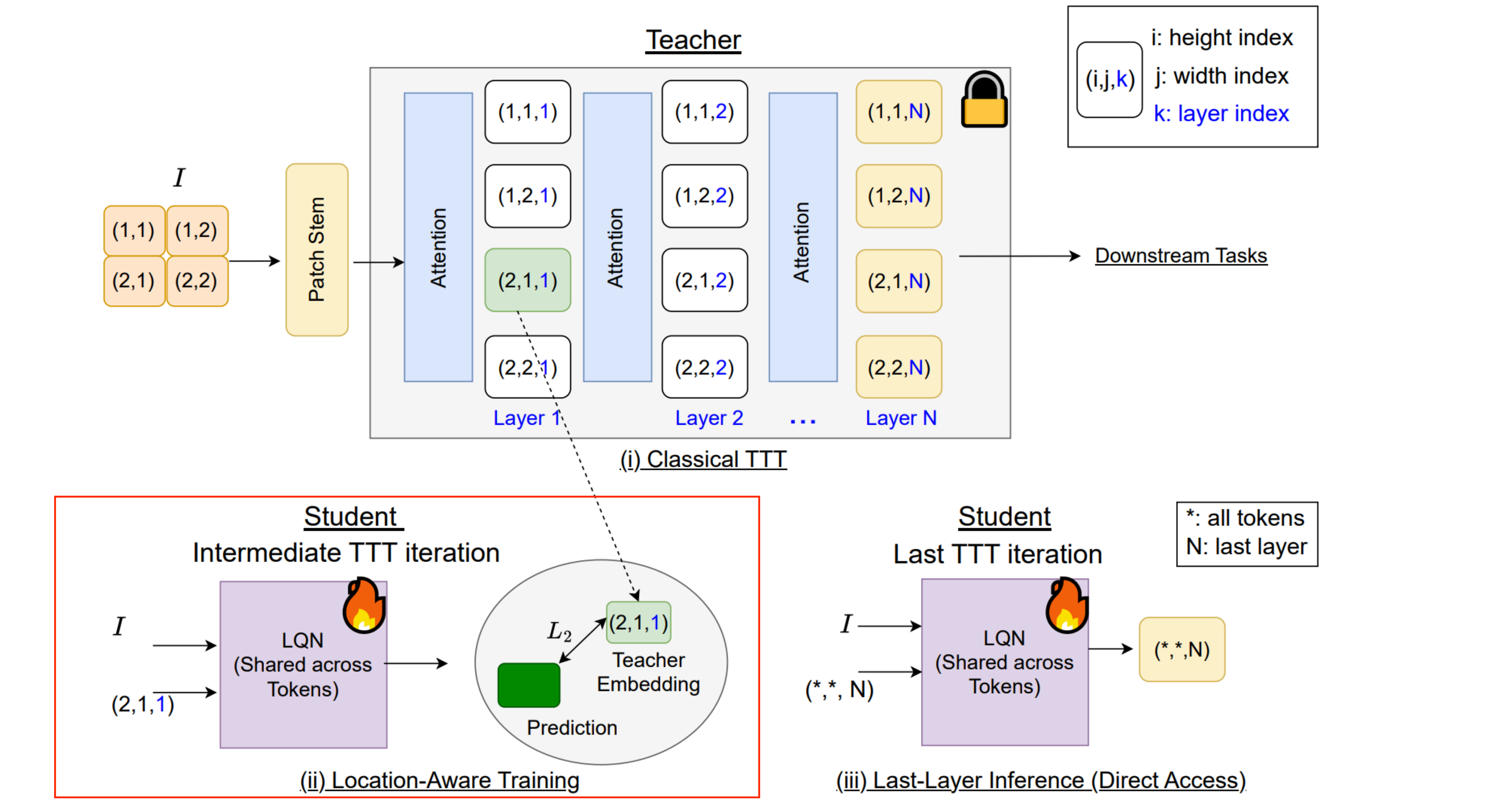
Seems like an impossible paradox :-)



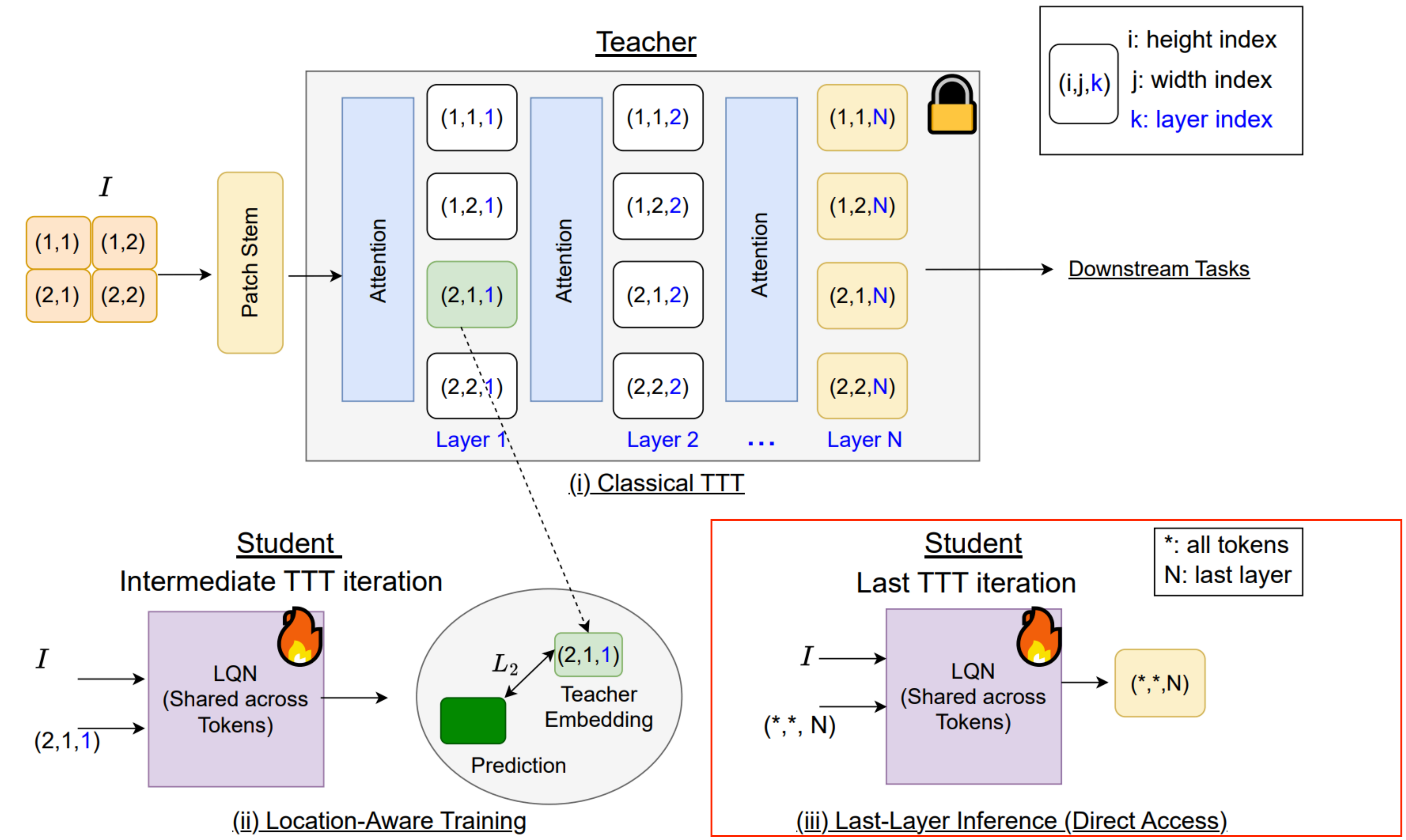
LAYER QUERY NETWORKS



LAYER QUERY NETWORKS



LAYER QUERY NETWORKS



Some important questions we could ask ourselves

- How does TTT help on classification
- What happens on Distribution Shifted datasets.
  - For eg, can the network adapt when it sees samples corrupted by variety of noises,
- On segmentation:
  - Can TTT show better qualitative results/quantitative numbers
- Are there any broader insights we could take away?

TTT on classification

Table 2: **LQN's Robustness to Natural Distribution Shifts**. CoOp and CoCoOp are tuned on ImageNet using 16-shot training data per category. Baseline CLIP, prompt ensemble, TPT, APM and LQN do not require training data. A ✓ in P means that method leveraged **pre-trained weights** on clean variant of train set aka, Image-net and downstream-ttt on corrupted version.

Method	P	ImageNet Top1 acc. ↑	ImageNet-A Top1 acc. ↑	ImageNet-V2 Top1 acc. ↑	ImageNet-R Top1 acc. ↑	ImageNet-Sketch Top1 acc. ↑	Average	OOD Average
CLIP-ViT-B/16(t)	✗	66.7	47.8	60.8	73.9	46.0	59.1	57.2
Ensemble	✗	68.3	49.8	61.8	77.6	48.2	61.2	59.4
TPT	✗	68.9	**54.7**	63.4	77.0	47.9	62.4	60.8
APM	✗	68.1	52.1	67.2	76.5	49.3	62.6	61.2
LQN (Two-Word) (Ours)	✗	68.7	53.2	67.8	77.1	50.1	63.4	61.8
LQN (3DLoc-Binded) (Ours)	✗	**69.4**	54.5	**68.5**	**78.0**	**51.0**	**64.3**	**62.7**
CoOp	✓	71.5	49.7	64.2	75.2	47.9	61.7	59.2
CoCoOp	✓	71.0	50.6	64.0	76.1	48.7	62.1	59.9
TPT + CoOp	✓	73.6	57.9	66.8	77.2	49.2	64.9	62.8
TPT + CoCoOp	✓	71.0	58.4	64.8	78.6	48.4	64.3	62.6
CLIP VIT-L/14(t)	✗	76.2	69.6	72.1	85.9	58.8	72.5	71.6
APM	✗	77.3	71.8	72.8	87.1	62.2	74.2	73.4
LQN (Two-Word) (Ours)	✗	77.9	73.0	73.6	88.2	63.0	75.1	74.3
LQN (3DLoc-Binded) (Ours)	✗	**78.6**	**74.2**	**74.3**	**89.1**	**64.1**	**76.1**	**75.3**
OpenCLIP-VIT-H/14(t)	✗	81.6	79.1	80.7	92.9	72.8	81.4	81.3
APM	✗	84.6	84.2	83.9	94.9	77.1	84.9	85.0
LQN (Two-Word) (Ours)	✗	85.2	85.0	84.7	95.5	78.0	85.7	85.7
LQN 3DLoc-Binded (Ours)	✗	**86.0**	**86.1**	**85.3**	**96.2**	**79.0**	**86.5**	**86.6**

TTT on classification

Table 2: **LQN's Robustness to Natural Distribution Shifts**. CoOp and CoCoOp are tuned on ImageNet using 16-shot training data per category. Baseline CLIP, prompt ensemble, TPT, APM and LQN do not require training data. A ✓ in P means that method leveraged **pre-trained weights** on clean variant of train set aka, Image-net and downstream-ttt on corrupted version.

Method	P	ImageNet Top1 acc. ↑	ImageNet-A Top1 acc. ↑	ImageNet-V2 Top1 acc. ↑	ImageNet-R Top1 acc. ↑	ImageNet-Sketch Top1 acc. ↑	Average	OOD Average
CLIP-ViT-B/16(t)	✗	66.7	47.8	60.8	73.9	46.0	59.1	57.2
Ensemble	✗	68.3	49.8	61.8	77.6	48.2	61.2	59.4
TPT	✗	68.9	**54.7**	63.4	77.0	47.9	62.4	60.8
APM	✗	68.1	52.1	67.2	76.5	49.3	62.6	61.2
LQN (Two-Word) (Ours)	✗	68.7	53.2	67.8	77.1	50.1	63.4	61.8
LQN (3DLoc-Binded) (Ours)	✗	**69.4**	54.5	**68.5**	**78.0**	**51.0**	**64.3**	**62.7**
CoOp	✓	71.5	49.7	64.2	75.2	47.9	61.7	59.2
CoCoOp	✓	71.0	50.6	64.0	76.1	48.7	62.1	59.9
TPT + CoOp	✓	73.6	57.9	66.8	77.2	49.2	64.9	62.8
TPT + CoCoOp	✓	71.0	58.4	64.8	78.6	48.4	64.3	62.6
CLIP VIT-L/14(t)	✗	76.2	69.6	72.1	85.9	58.8	72.5	71.6
APM	✗	77.3	71.8	72.8	87.1	62.2	74.2	73.4
LQN (Two-Word) (Ours)	✗	77.9	73.0	73.6	88.2	63.0	75.1	74.3
LQN (3DLoc-Binded) (Ours)	✗	**78.6**	**74.2**	**74.3**	**89.1**	**64.1**	**76.1**	**75.3**
OpenCLIP-VIT-H/14(t)	✗	81.6	79.1	80.7	92.9	72.8	81.4	81.3
APM	✗	84.6	84.2	83.9	94.9	77.1	84.9	85.0
LQN (Two-Word) (Ours)	✗	85.2	85.0	84.7	95.5	78.0	85.7	85.7
LQN 3DLoc-Binded (Ours)	✗	**86.0**	**86.1**	**85.3**	**96.2**	**79.0**	**86.5**	**86.6**

TTT on classification

Table 2: **LQN's Robustness to Natural Distribution Shifts**. CoOp and CoCoOp are tuned on ImageNet using 16-shot training data per category. Baseline CLIP, prompt ensemble, TPT, APM and LQN do not require training data. A ✓ in P means that method leveraged **pre-trained weights** on clean variant of train set aka, Image-net and downstream-ttt on corrupted version.

Method	P	ImageNet Top1 acc. ↑	ImageNet-A Top1 acc. ↑	ImageNet-V2 Top1 acc. ↑	ImageNet-R Top1 acc. ↑	ImageNet-Sketch Top1 acc. ↑	Average	OOD Average
CLIP-ViT-B/16(t)	✗	66.7	47.8	60.8	73.9	46.0	59.1	57.2
Ensemble	✗	68.3	49.8	61.8	77.6	48.2	61.2	59.4
TPT	✗	68.9	**54.7**	63.4	77.0	47.9	62.4	60.8
APM	✗	68.1	52.1	67.2	76.5	49.3	62.6	61.2
LQN (Two-Word) (Ours)	✗	68.7	53.2	67.8	77.1	50.1	63.4	61.8
LQN (3DLoc-Binded) (Ours)	✗	**69.4**	54.5	**68.5**	**78.0**	**51.0**	**64.3**	**62.7**
CoOp	✓	71.5	49.7	64.2	75.2	47.9	61.7	59.2
CoCoOp	✓	71.0	50.6	64.0	76.1	48.7	62.1	59.9
TPT + CoOp	✓	73.6	57.9	66.8	77.2	49.2	64.9	62.8
TPT + CoCoOp	✓	71.0	58.4	64.8	78.6	48.4	64.3	62.6
CLIP VIT-L/14(t)	✗	76.2	69.6	72.1	85.9	58.8	72.5	71.6
APM	✗	77.3	71.8	72.8	87.1	62.2	74.2	73.4
LQN (Two-Word) (Ours)	✗	77.9	73.0	73.6	88.2	63.0	75.1	74.3
LQN (3DLoc-Binded) (Ours)	✗	**78.6**	**74.2**	**74.3**	**89.1**	**64.1**	**76.1**	**75.3**
OpenCLIP-VIT-H/14(t)	✗	81.6	79.1	80.7	92.9	72.8	81.4	81.3
APM	✗	84.6	84.2	83.9	94.9	77.1	84.9	85.0
LQN (Two-Word) (Ours)	✗	85.2	85.0	84.7	95.5	78.0	85.7	85.7
LQN 3DLoc-Binded (Ours)	✗	**86.0**	**86.1**	**85.3**	**96.2**	**79.0**	**86.5**	**86.6**

TTT on classification

Table 2: **LQN's Robustness to Natural Distribution Shifts**. CoOp and CoCoOp are tuned on ImageNet using 16-shot training data per category. Baseline CLIP, prompt ensemble, TPT, APM and LQN do not require training data. A ✓ in P means that method leveraged **pre-trained weights** on clean variant of train set aka, Image-net and downstream-ttt on corrupted version.

Method	P	ImageNet Top1 acc. ↑	ImageNet-A Top1 acc. ↑	ImageNet-V2 Top1 acc. ↑	ImageNet-R Top1 acc. ↑	ImageNet-Sketch Top1 acc. ↑	Average	OOD Average
CLIP-ViT-B/16(t)	✗	66.7	47.8	60.8	73.9	46.0	59.1	57.2
Ensemble	✗	68.3	49.8	61.8	77.6	48.2	61.2	59.4
TPT	✗	68.9	**54.7**	63.4	77.0	47.9	62.4	60.8
APM	✗	68.1	52.1	67.2	76.5	49.3	62.6	61.2
LQN (Two-Word) (Ours)	✗	68.7	53.2	67.8	77.1	50.1	63.4	61.8
LQN (3DLoc-Binded) (Ours)	✗	**69.4**	54.5	**68.5**	**78.0**	**51.0**	**64.3**	**62.7**
CoOp	✓	71.5	49.7	64.2	75.2	47.9	61.7	59.2
CoCoOp	✓	71.0	50.6	64.0	76.1	48.7	62.1	59.9
TPT + CoOp	✓	73.6	57.9	66.8	77.2	49.2	64.9	62.8
TPT + CoCoOp	✓	71.0	58.4	64.8	78.6	48.4	64.3	62.6
CLIP VIT-L/14(t)	✗	76.2	69.6	72.1	85.9	58.8	72.5	71.6
APM	✗	77.3	71.8	72.8	87.1	62.2	74.2	73.4
LQN (Two-Word) (Ours)	✗	77.9	73.0	73.6	88.2	63.0	75.1	74.3
LQN (3DLoc-Binded) (Ours)	✗	**78.6**	**74.2**	**74.3**	**89.1**	**64.1**	**76.1**	**75.3**
OpenCLIP-VIT-H/14(t)	✗	81.6	79.1	80.7	92.9	72.8	81.4	81.3
APM	✗	84.6	84.2	83.9	94.9	77.1	84.9	85.0
LQN (Two-Word) (Ours)	✗	85.2	85.0	84.7	95.5	78.0	85.7	85.7
LQN 3DLoc-Binded (Ours)	✗	**86.0**	**86.1**	**85.3**	**96.2**	**79.0**	**86.5**	**86.6**

TTT on classification

Table 2: **LQN's Robustness to Natural Distribution Shifts**. CoOp and CoCoOp are tuned on ImageNet using 16-shot training data per category. Baseline CLIP, prompt ensemble, TPT, APM and LQN do not require training data. A ✓ in P means that method leveraged **pre-trained weights** on clean variant of train set aka, Image-net and downstream-ttt on corrupted version.

Method	P	ImageNet Top1 acc. ↑	ImageNet-A Top1 acc. ↑	ImageNet-V2 Top1 acc. ↑	ImageNet-R Top1 acc. ↑	ImageNet-Sketch Top1 acc. ↑	Average	OOD Average
CLIP-ViT-B/16(t)	✗	66.7	47.8	60.8	73.9	46.0	59.1	57.2
Ensemble	✗	68.3	49.8	61.8	77.6	48.2	61.2	59.4
TPT	✗	68.9	**54.7**	63.4	77.0	47.9	62.4	60.8
APM	✗	68.1	52.1	67.2	76.5	49.3	62.6	61.2
LQN (Two-Word) (Ours)	✗	68.7	53.2	67.8	77.1	50.1	63.4	61.8
LQN (3DLoc-Binded) (Ours)	✗	**69.4**	54.5	**68.5**	**78.0**	**51.0**	**64.3**	**62.7**
CoOp	✓	71.5	49.7	64.2	75.2	47.9	61.7	59.2
CoCoOp	✓	71.0	50.6	64.0	76.1	48.7	62.1	59.9
TPT + CoOp	✓	73.6	57.9	66.8	77.2	49.2	64.9	62.8
TPT + CoCoOp	✓	71.0	58.4	64.8	78.6	48.4	64.3	62.6
CLIP VIT-L/14(t)	✗	76.2	69.6	72.1	85.9	58.8	72.5	71.6
APM	✗	77.3	71.8	72.8	87.1	62.2	74.2	73.4
LQN (Two-Word) (Ours)	✗	77.9	73.0	73.6	88.2	63.0	75.1	74.3
LQN (3DLoc-Binded) (Ours)	✗	**78.6**	**74.2**	**74.3**	**89.1**	**64.1**	**76.1**	**75.3**
OpenCLIP-VIT-H/14(t)	✗	81.6	79.1	80.7	92.9	72.8	81.4	81.3
APM	✗	84.6	84.2	83.9	94.9	77.1	84.9	85.0
LQN (Two-Word) (Ours)	✗	85.2	85.0	84.7	95.5	78.0	85.7	85.7
LQN 3DLoc-Binded (Ours)	✗	**86.0**	**86.1**	**85.3**	**96.2**	**79.0**	**86.5**	**86.6**

TTT for semantic segmentation

Table 6: **LQN for semantic segmentation.** †On ADE20K, these models resize the shortest side of images to the indicated scale during inference, while preserving the aspect ratio. ‡Re-implementation by EoMT. ViT-Adapter + Mask2Former and EoMT use windowed inference, dividing each image into multiple crops. DA is Depth Anything [42]. Methods marked with § use TTT. LQN obtains higher performance than Mask2former+ VIT-Adapter-L with lower GFlops.

Method	Backbone	Pre-training	Params	Cityscapes *val* [?]			ADE20K *val* [26]		
				Input size	GFLOPs	mIoU	Input size	GFLOPs	mIoU
Mask2Former† [33]	Swin-L [34]	IN21K	216M	1024 × 2048	–	83.3	640^2	–	56.1
MaskDINO† [39]	Swin-L [34]	IN21K	223M	–	–	–	640^2	–	56.6
OneFormer† [37]	ConvNext-XL [36]	IN21K	373M	1024 × 2048	775	83.6	640^2	607	57.4
OneFormer† [37]	DiNAT-L [38]	IN21K	223M	1024 × 2048	450	83.1	896^2	678	58.1
kMaX-DeepLab [35]	ConvNext-L [36]	IN21K	232M	1025 × 2049	1673	83.5	–	–	–
Mask2Former [33]	ViT-L [17]	DINOv2 + DA	–	896 × 1792	–	84.8	896^2	–	59.4
Mask2Former‡ [33]	ViT-Adapter-L‡ [40]	DINOv2	351M	1024^2	5200	84.5	512^2	910	58.9
EoMT(t)	ViT-L [17]	DINOv2	319M	1024^2	4350	84.2	512^2	721	58.4
APM§	MLP [41]	DINOv2	350M	1024^2	4540	85.1	512^2	911	58.8
LQN-3DBinded(Ours)§	MLP [41]	DINOv2	350M	1024^2	4490	**85.7**	512^2	861	**61.2**

TTT for semantic segmentation

Table 6: **LQN for semantic segmentation.** †On ADE20K, these models resize the shortest side of images to the indicated scale during inference, while preserving the aspect ratio. ‡Re-implementation by EoMT. ViT-Adapter + Mask2Former and EoMT use windowed inference, dividing each image into multiple crops. DA is Depth Anything [42]. Methods marked with § use TTT. LQN obtains higher performance than Mask2former+ VIT-Adapter-L with lower GFlops.

Method	Backbone	Pre-training	Params	Cityscapes *val* [?]			ADE20K *val* [26]		
				Input size	GFLOPs	mIoU	Input size	GFLOPs	mIoU
Mask2Former† [33]	Swin-L [34]	IN21K	216M	1024×2048	–	83.3	640^2	–	56.1
MaskDINO† [39]	Swin-L [34]	IN21K	223M	–	–	–	640^2	–	56.6
OneFormer† [37]	ConvNext-XL [36]	IN21K	373M	1024×2048	775	83.6	640^2	607	57.4
OneFormer† [37]	DiNAT-L [38]	IN21K	223M	1024×2048	450	83.1	896^2	678	58.1
kMaX-DeepLab [35]	ConvNext-L [36]	IN21K	232M	1025×2049	1673	83.5	–	–	–
Mask2Former [33]	ViT-L [17]	DINOv2 + DA	–	896×1792	–	84.8	896^2	–	59.4
Mask2Former‡ [33]	ViT-Adapter-L‡ [40]	DINOv2	351M	1024^2	5200	84.5	512^2	910	58.9
EoMT(t)	ViT-L [17]	DINOv2	319M	1024^2	4350	84.2	512^2	721	58.4
APM§	MLP [41]	DINOv2	350M	1024^2	4540	85.1	512^2	911	58.8
LQN-3DBinded(Ours)§	MLP [41]	DINOv2	350M	1024^2	4490	**85.7**	512^2	861	**61.2**

TTT for semantic segmentation

Table 6: **LQN for semantic segmentation.** †On ADE20K, these models resize the shortest side of images to the indicated scale during inference, while preserving the aspect ratio. ‡Re-implementation by EoMT. ViT-Adapter + Mask2Former and EoMT use windowed inference, dividing each image into multiple crops. DA is Depth Anything [42]. Methods marked with § use TTT. LQN obtains higher performance than Mask2former+ VIT-Adapter-L with lower GFlops.

Method	Backbone	Pre-training	Params	Cityscapes *val* [?]			ADE20K *val* [26]		
				Input size	GFLOPs	mIoU	Input size	GFLOPs	mIoU
Mask2Former† [33]	Swin-L [34]	IN21K	216M	1024 × 2048	–	83.3	640^2	–	56.1
MaskDINO† [39]	Swin-L [34]	IN21K	223M	–	–	–	640^2	–	56.6
OneFormer† [37]	ConvNext-XL [36]	IN21K	373M	1024 × 2048	775	83.6	640^2	607	57.4
OneFormer† [37]	DiNAT-L [38]	IN21K	223M	1024 × 2048	450	83.1	896^2	678	58.1
kMaX-DeepLab [35]	ConvNext-L [36]	IN21K	232M	1025 × 2049	1673	83.5	–	–	–
Mask2Former [33]	ViT-L [17]	DINOv2 + DA	–	896 × 1792	–	84.8	896^2	–	59.4
Mask2Former‡ [33]	ViT-Adapter-L‡ [40]	DINOv2	351M	1024^2	5200	84.5	512^2	910	58.9
EoMT(t)	ViT-L [17]	DINOv2	319M	1024^2	4350	84.2	512^2	721	58.4
APM§	MLP [41]	DINOv2	350M	1024^2	4540	85.1	512^2	911	58.8
LQN-3DBinded(Ours)§	MLP [41]	DINOv2	350M	1024^2	4490	**85.7**	512^2	861	**61.2**

TTT for semantic segmentation

Table 6: **LQN for semantic segmentation.** †On ADE20K, these models resize the shortest side of images to the indicated scale during inference, while preserving the aspect ratio. ‡Re-implementation by EoMT. ViT-Adapter + Mask2Former and EoMT use windowed inference, dividing each image into multiple crops. DA is Depth Anything [42]. Methods marked with § use TTT. LQN obtains higher performance than Mask2former+ VIT-Adapter-L with lower GFlops.

Method	Backbone	Pre-training	Params	Cityscapes *val* [?]			ADE20K *val* [26]		
				Input size	GFLOPs	mIoU	Input size	GFLOPs	mIoU
Mask2Former† [33]	Swin-L [34]	IN21K	216M	1024×2048	–	83.3	640^2	–	56.1
MaskDINO† [39]	Swin-L [34]	IN21K	223M	–	–	–	640^2	–	56.6
OneFormer† [37]	ConvNext-XL [36]	IN21K	373M	1024×2048	775	83.6	640^2	607	57.4
OneFormer† [37]	DiNAT-L [38]	IN21K	223M	1024×2048	450	83.1	896^2	678	58.1
kMaX-DeepLab [35]	ConvNext-L [36]	IN21K	232M	1025×2049	1673	83.5	–	–	–
Mask2Former [33]	ViT-L [17]	DINOv2 + DA	–	896×1792	–	84.8	896^2	–	59.4
Mask2Former‡ [33]	ViT-Adapter-L‡ [40]	DINOv2	351M	1024^2	5200	84.5	512^2	910	58.9
EoMT(t)	ViT-L [17]	DINOv2	319M	1024^2	4350	84.2	512^2	721	58.4
APM§	MLP [41]	DINOv2	350M	1024^2	4540	85.1	512^2	911	58.8
LQN-3DBinded(Ours)§	MLP [41]	DINOv2	350M	1024^2	4490	**85.7**	512^2	861	**61.2**

TTT for semantic segmentation

- TTT over the baseline model (Maskformer) improves performance.
  - But it is adds more computational cost.
  - This is still lower than fancier segmentation architectures.

- Any broader insights we could take away?

The notion of Direct Memory Access

Arch	Spatial	Depth	D.A.
CNN[18]/Trans.[19]	✓	×	×
Universal Transf.[20]	✓	✓	×
LQN (Ours)	✓	✓	✓

Table 1: Comparison of existing nets by weight sharing across spatial-inputs, depth and the nature of computation. D.A: Direct memory access[21], i.e. the ability to access layers at any depth in a constant amount of time, without calculating previous layers.

- Any broader insights we could take away?

The notion of Direct Memory Access

Arch	Spatial	Depth	D.A.
CNN[18]/Trans.[19]	✓	×	×
Universal Transf.[20]	✓	✓	×
LQN (Ours)	✓	✓	✓

Table 1: Comparison of existing nets by weight sharing across spatial-inputs, depth and the nature of computation. D.A: Direct memory access[21], i.e. the ability to access layers at any depth in a constant amount of time, without calculating previous layers.

- Any broader insights we could take away?

The notion of Direct Memory Access

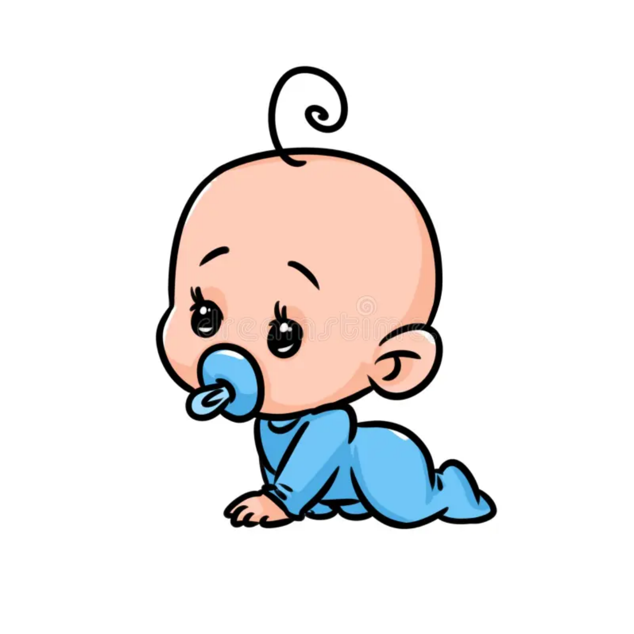
Arch	Spatial	Depth	D.A.
CNN[18]/Trans.[19]	✓	×	×
Universal Transf.[20]	✓	✓	×
LQN (Ours)	✓	✓	✓

Table 1: Comparison of existing nets by weight sharing across spatial-inputs, depth and the nature of computation. D.A: Direct memory access[21], i.e. the ability to access layers at any depth in a constant amount of time, without calculating previous layers.

- Neural net can decide its own depth.
- It could “unroll” depth, and pause processing when done (early exit).

- In LQN, it will just query what is at a particular depth, and directly get the answer.

What our community has done till now



- Vit base

12 blocks

- Vit large

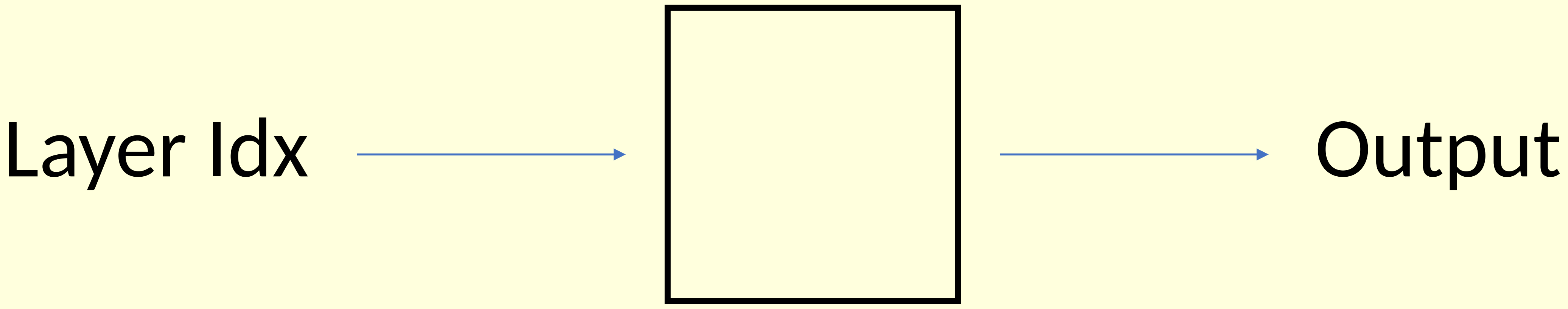
24 blocks

- Vit huge

32 blocks

Each network has separate weights.

Depth As an Interpolation Dimension



- Suppose you trained this network to predict upto 12 layers
- You could query it from layer 13… 24 onwards.
- Will it generalize to layers it has never been trained for?
- How well it would perform?

Depth As an Interpolation Dimension

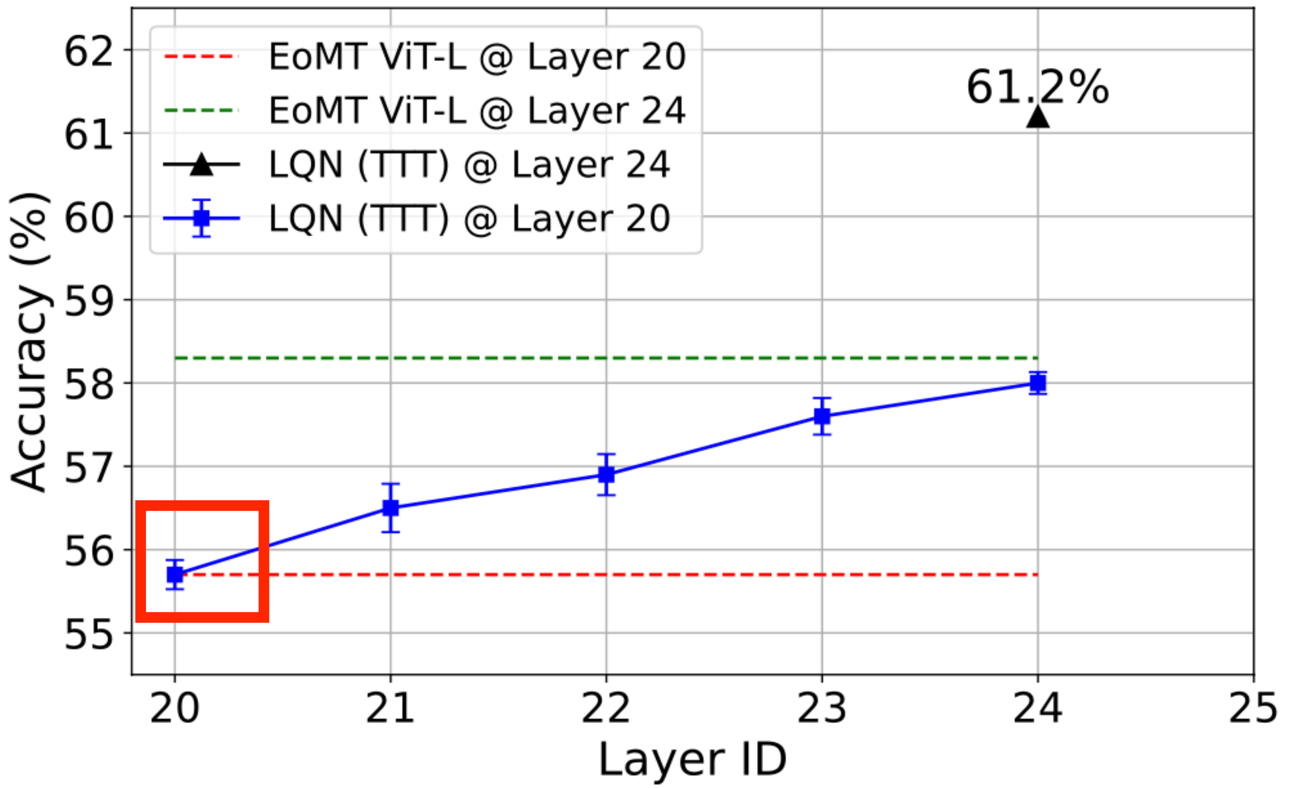


Figure 1: **LQN generalizes to layers which were not seen during training:** LQN trained on 20 layers of VIT-L for semantic segmentation on ADE20K demonstrates increasing performance as features are queried from the the last 4 layers.

Depth As an Interpolation Dimension

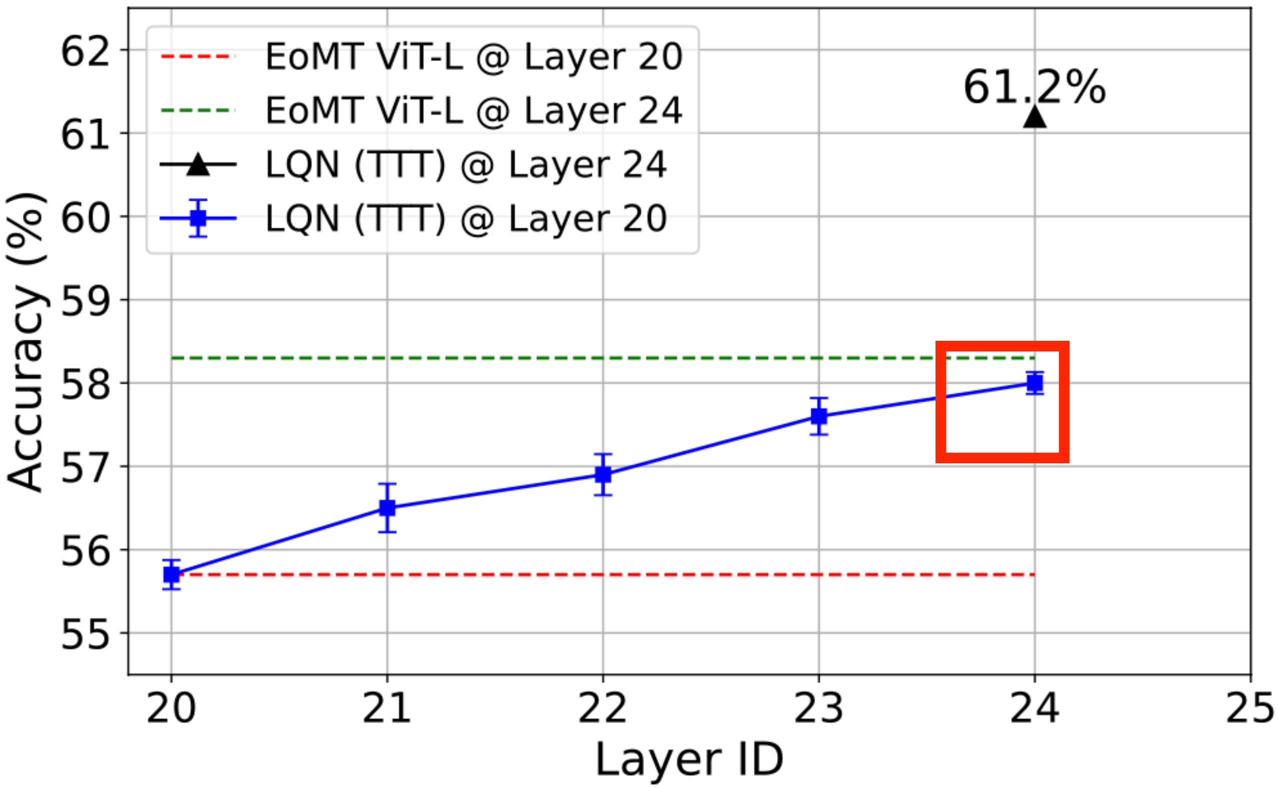


Figure 1: **LQN generalizes to layers which were not seen during training:** LQN trained on 20 layers of VIT-L for semantic segmentation on ADE20K demonstrates increasing performance as features are queried from the the last 4 layers.

Depth As an Interpolation Dimension

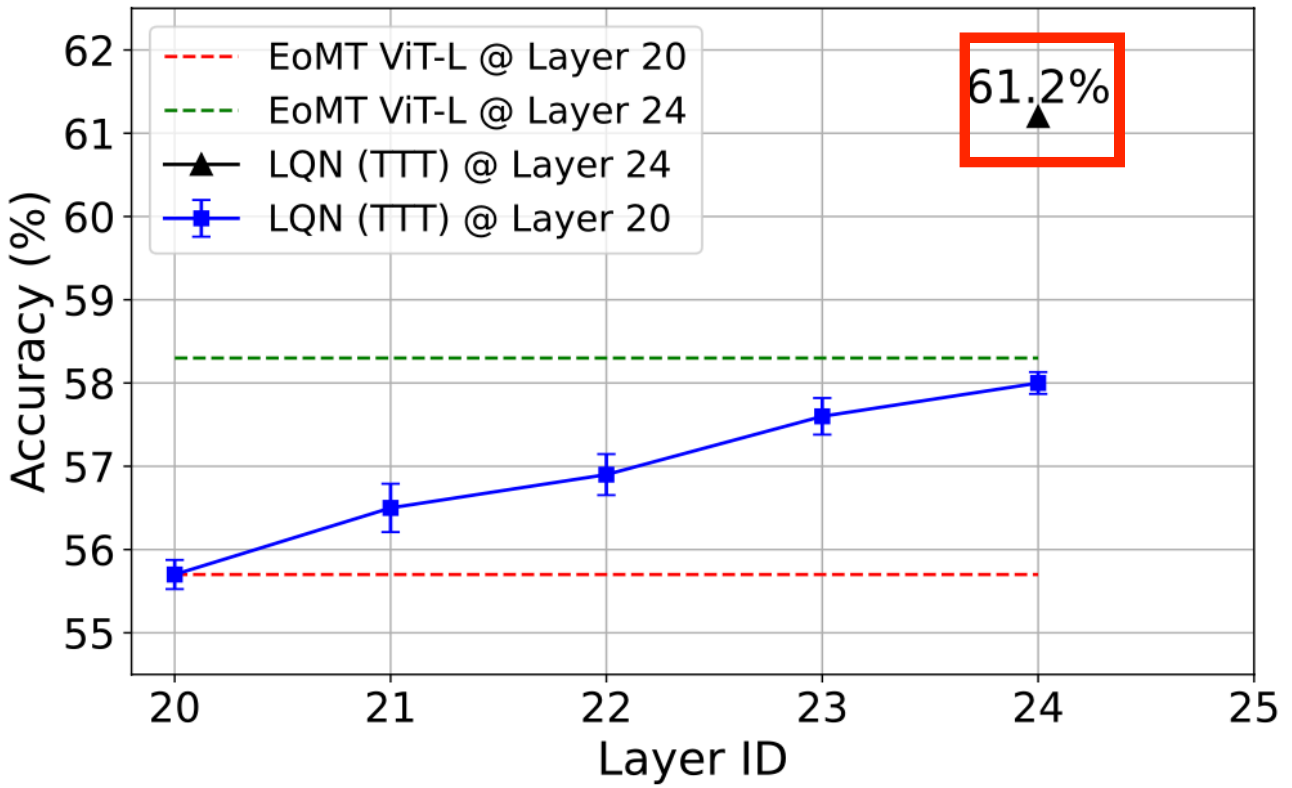
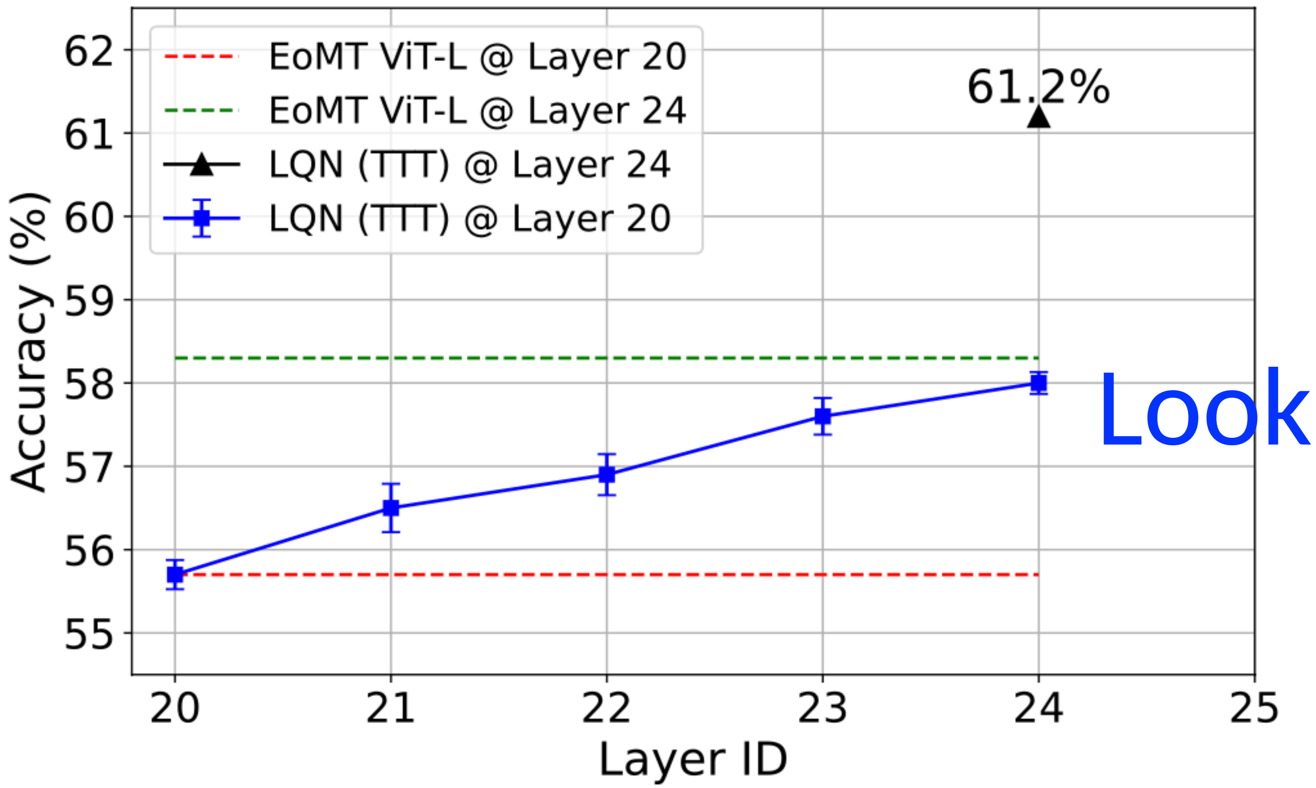


Figure 1: **LQN generalizes to layers which were not seen during training:** LQN trained on 20 layers of VIT-L for semantic segmentation on ADE20K demonstrates increasing performance as features are queried from the the last 4 layers.

Depth As an Interpolation Dimension



Look at Blue Line

Figure 1: **LQN generalizes to layers which were not seen during training:** LQN trained on 20 layers of VIT-L for semantic segmentation on ADE20K demonstrates increasing performance as features are queried from the the last 4 layers.

How can one generalize across depths?

- Its not surprising.
  - Transformers can generalize to sequence lengths not seen during training.
- LQN's insight: Depth is also an additional spatial dimensions
  - So you can generalize to depths not seen during training.

Sequential Decoding vs O(1) decoding

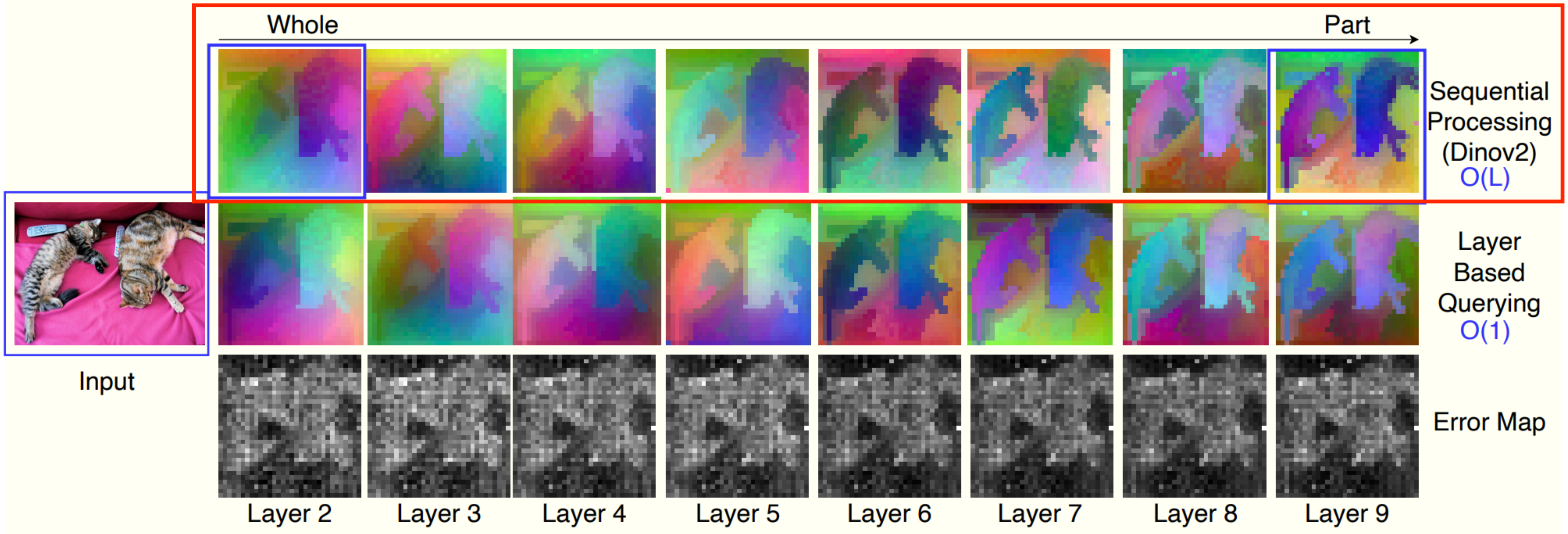


Figure 3: **LQN's feature analysis (Top Row):** t-SNE visualization of intermediate features as one traverses different layers of a teacher (eg. DinoV2[32]). Sequential processing here takes O(L), where L is the layer depth. **(Middle Row):** Predicted features from LQN (ours). Layer-based querying yields any layer's features in constant time irrespective of layer depth. **(Bottom Row):** L_2 error map between two feature maps. As we go deeper, the error decreases.

Sequential Decoding vs O(1) decoding

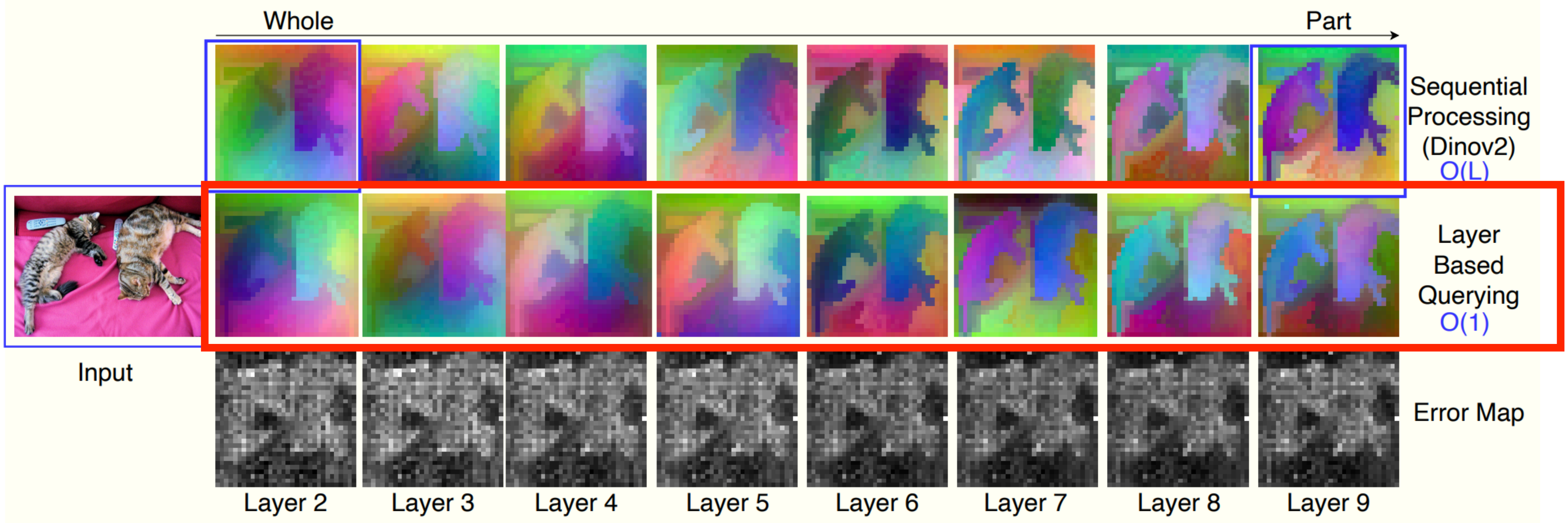


Figure 3: **LQN's feature analysis (Top Row):** t-SNE visualization of intermediate features as one traverses different layers of a teacher (eg. DinoV2[32]). Sequential processing here takes O(L), where L is the layer depth. **(Middle Row):** Predicted features from LQN (ours). Layer-based querying yields any layer's features in constant time irrespective of layer depth. **(Bottom Row):** L_2 error map between two feature maps. As we go deeper, the error decreases.

Sequential Decoding vs O(1) decoding

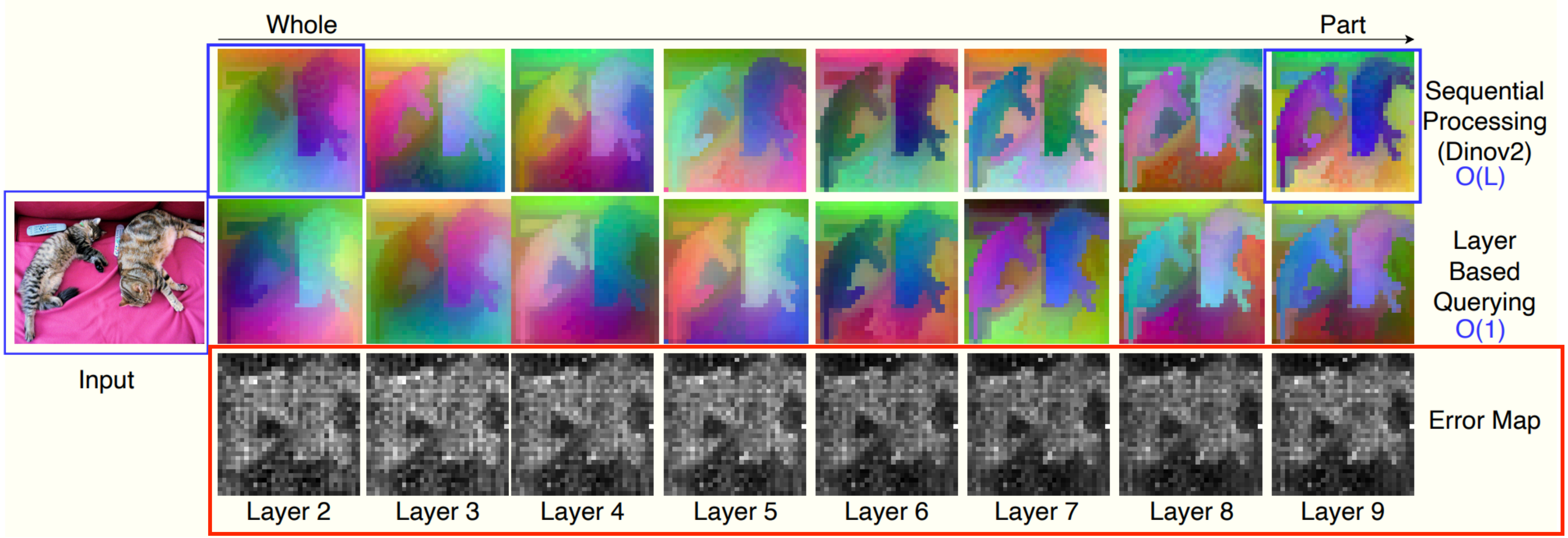


Figure 3: **LQN's feature analysis (Top Row):** t-SNE visualization of intermediate features as one traverses different layers of a teacher (eg. DinoV2[32]). Sequential processing here takes O(L), where L is the layer depth. **(Middle Row):** Predicted features from LQN (ours). Layer-based querying yields any layer's features in constant time irrespective of layer depth. **(Bottom Row):** L_2 error map between two feature maps. As we go deeper, the error decreases.

Constant Time Inference

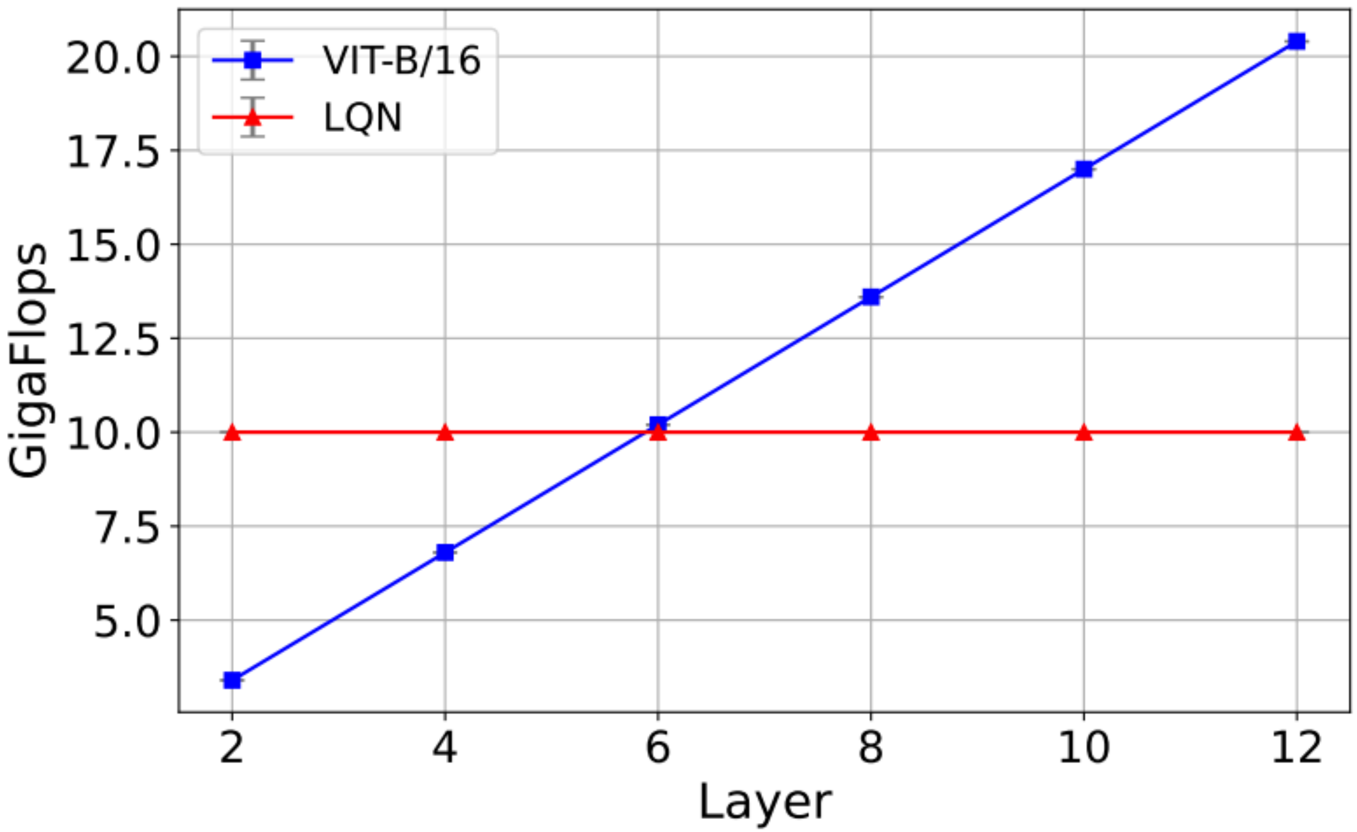


Figure 4: GFlops as a function of layer depth in a transformer-based model like ViT-B/16, vs LQN.

- Deep Learning as we know it involves stacking blocks over one another
- Another way is to query the feature at particular depth and directly ask for it.
  - No need of layer stacking.
- It might help make deep learning faster.

Thank You

Takeaway: Depth can be “queried” instead of stacking-blocks.