# Which Transformer to Favor: A Comparative Analysis of Efficiency in Vision Transformers

Tobias Nauen (RPTU, DFKI), Sebastian Palacio (ABB),
Federico Raue (DFKI), Andreas Dengel (DFKI, RPTU)

# There are many efficient transformer variants for computer vision.

- Transformers are the state-of-the-art models in computer vision.
- Their $O(N^2)$ computational complexity makes handling high resolution images expensive.

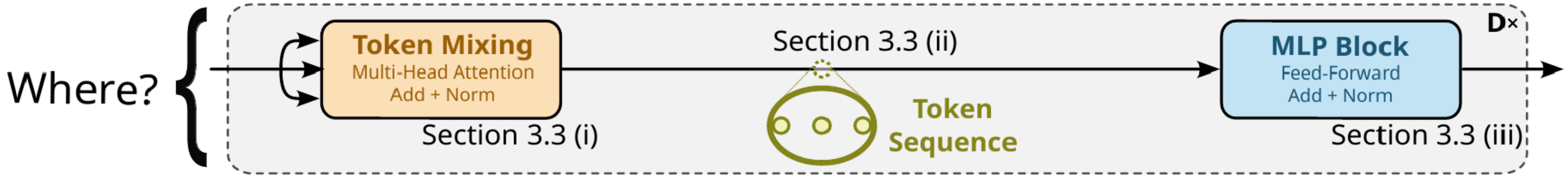
Routing Transformer
HaloNet EfficientFormer
Scatterbrain
PatchConvNet
Flash Attention FocalNet
MedViT ViT Synthesizer
Swin XCiT
Sinkhorn Transformer
EViT CaiT Linformer
ToMe MLP Mixer
Switch Transformer
GFNet DeiT FNet
Nyströmformer
Performer

# There are many efficient transformer variants for computer vision.

- Transformers are the state-of-the-art models in computer vision.
- Their $O(N^2)$ computational complexity makes handling high resolution images expensive.
- Many efficient transformers introduced in the literature.

Routing Transformer
HaloNet EfficientFormer
Scatterbrain
PatchConvNet
Flash Attention FocalNet
MedViT ViT Synthesizer
Swin XCiT
Sinkhorn Transformer
EViT CaiT Linformer
ToMe MLP Mixer
Switch Transformer
GFNet DeiT FNet
Nyströmformer
Performer

# Efficient transformers change the architecture in three major places.




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

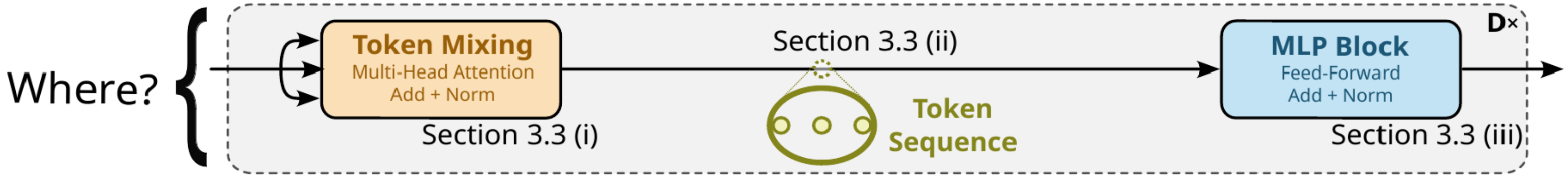
**Token Sequence**

- Token Removal
- Token Merging
- Summary Tokens

**MLP Block**

- More MLPs

# Efficient transformers change the architecture in three major places.



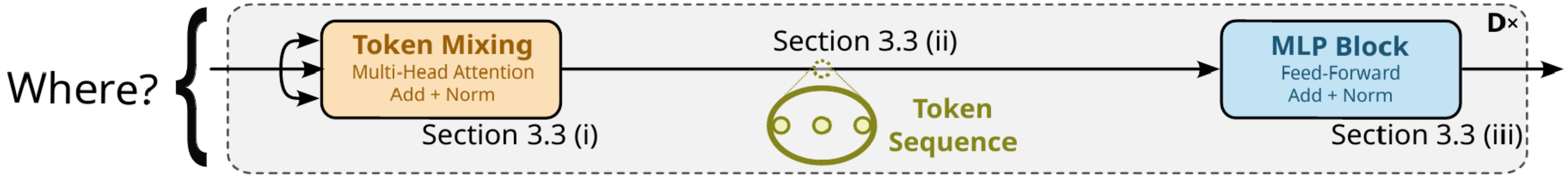

**Token Mixing**
- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

Exploit the low-rank matrix $QK^\top$

**Token Sequence**
- …oval
- …ging
- Summary Tokens

**MLP Block**
- More MLPs

# Efficient transformers change the architecture in three major places.




**Token Mixing**
- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

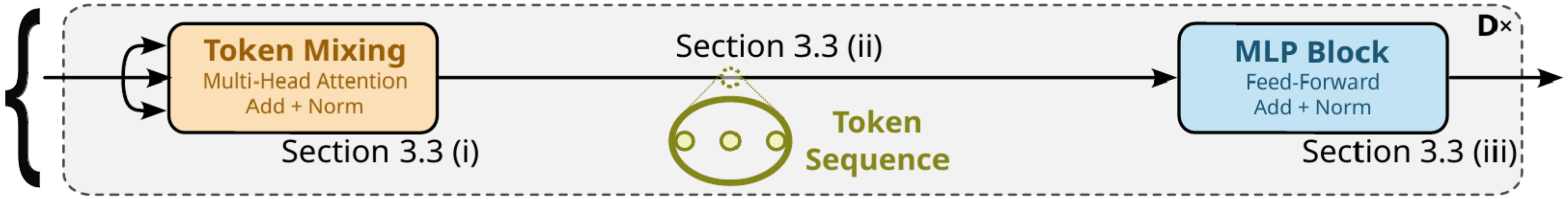
Exploit that many entries of the attention matrix are ≈0

**T en Sequence**
- emoval
- erging
- ry Tokens

**MLP Block**
- More MLPs

# Efficient transformers change the architecture in three major places.

Where?




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention ⇒ Set a fixed attention pattern
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

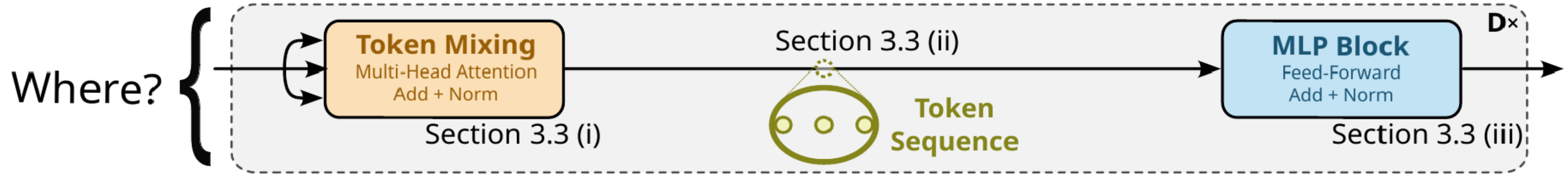
**Token Sequence**

- Token Removal
- Merging
- ary Tokens

**MLP Block**

- More MLPs

# Efficient transformers change the architecture in three major places.




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

Shift the activation from $QK^\top$ to $Q$ and $K$ individually

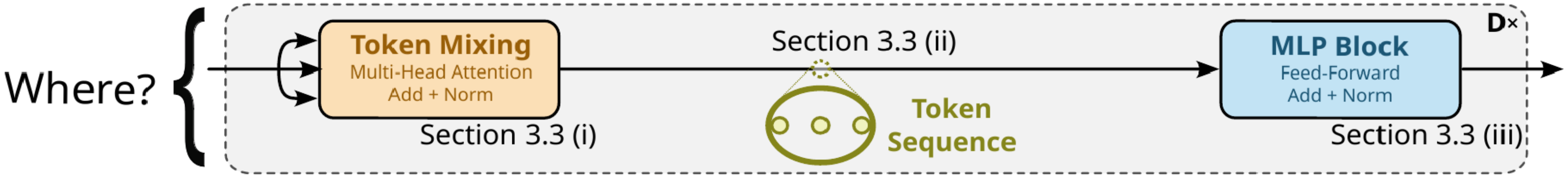
**Token Sequence**

- Token Removal
- Token Merging
- ry Tokens

**MLP Block**

- More MLPs

# Efficient transformers change the architecture in three major places.

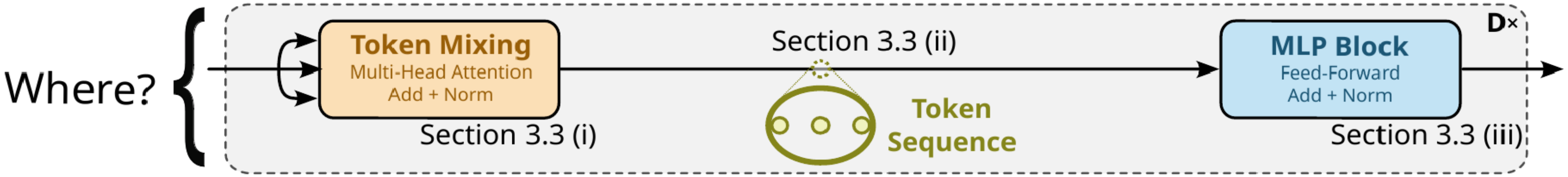



**Token Mixing**
- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention ⇨ Use convolutions in the attention calculation
- Fourier Attention
- Non-Attention Shuffling

**Token Sequence**
- Token Removal
- Token Merging
- Summary Tokens

**MLP Block**
- More MLPs

# Efficient transformers change the architecture in three major places.

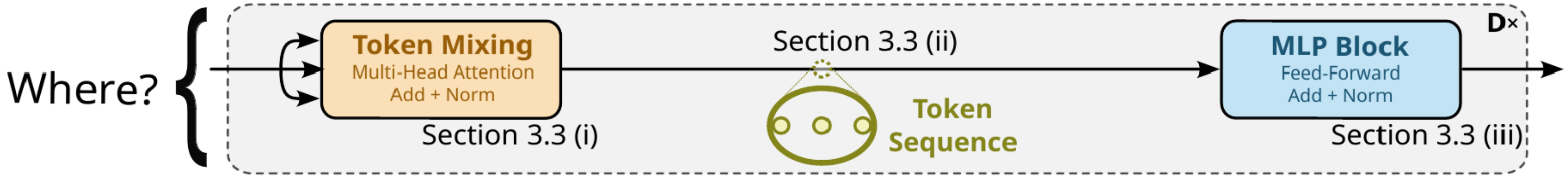



**Token Mixing**
- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention ⇨ Utilize the FFT
- Non-Attention Shuffling

**Token Sequence**
- Token Removal
- Token Merging
- Summary Tokens

**MLP Block**
- More MLPs

# Efficient transformers change the architecture in three major places.




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling ⇨ Use a different mechanism to mix the token-information

**Token Sequence**

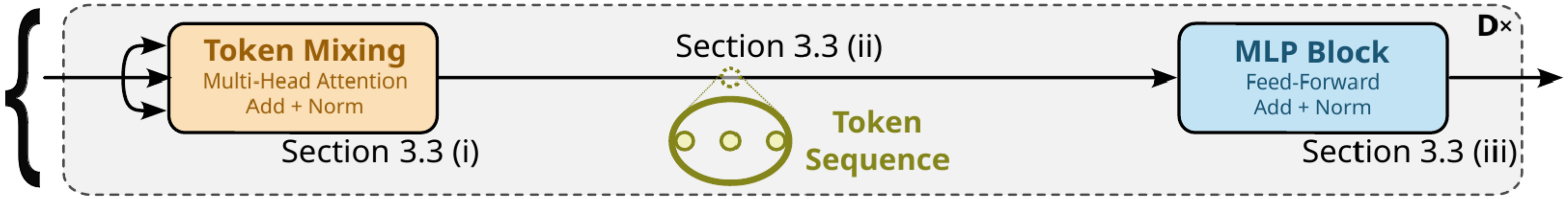
- Token Removal
- Token Merging
- Summary Tokens

**MLP Block**

- More MLPs

# Efficient transformers change the architecture in three major places.

Where?




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

**Token Sequen**

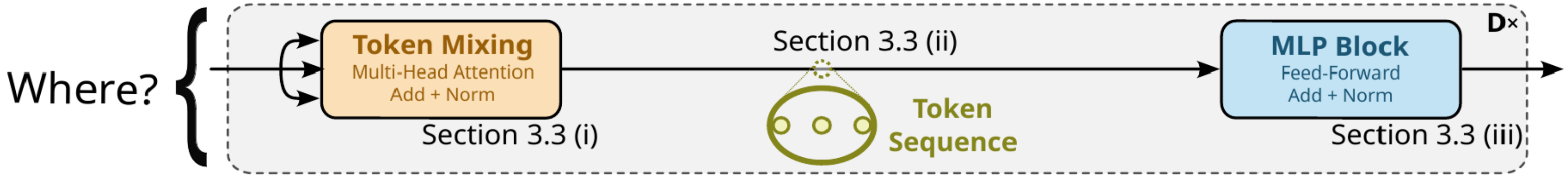
- Token Removal
- Token Merging
- Summary Tokens

Remove uninformative tokens

**MLP Block**

ILPs

# Efficient transformers change the architecture in three major places.




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

**Token Sequence**

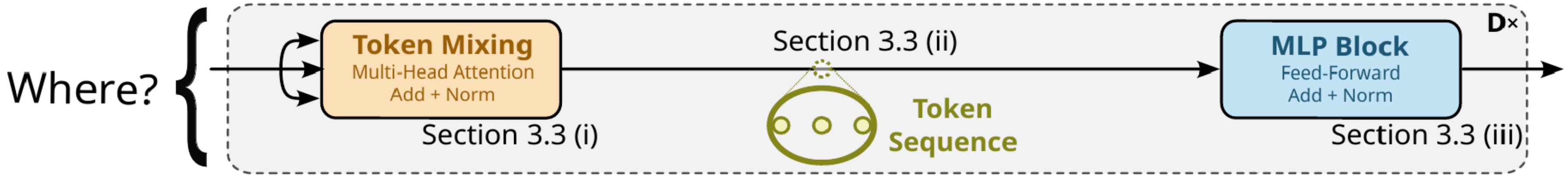
- Token Removal
- Token Merging
- Summary Tokens

Merge redundant tokens

**MLP Block**

MLPs

# Efficient transformers change the architecture in three major places.




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

**Token Sequence**

- Token Removal
- Token Merging
- Summary Tokens

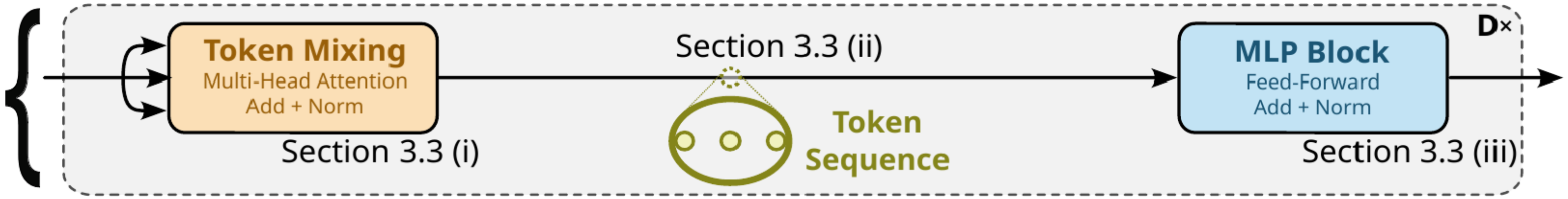
Summarize sets of tokens into fewer new tokens

**MLP Block**

- More MLPs

# Efficient transformers change the architecture in three major places.

Where?




**Token Mixing**

- Low-Rank Attention
- Sparse Attention
- Fixed Attention
- Kernel Attention
- Hybrid Attention
- Fourier Attention
- Non-Attention Shuffling

**Token Sequence**

- Token Removal
- Token Merging
- Summary Tokens

**MLP Bl**

- More MLPs ⇨

Shift calculations to the MLP block

# How can we find the best efficient transformer?

- Which modifications and overall strategies are the most efficient?
- Are these modifications worth considering over the baseline ViT?
- What other dimensions influence efficiency?

# How can we find the best efficient transformer?

- Which modifications and overall strategies are the most efficient?
- Are these modifications worth considering over the baseline ViT?
- What other dimensions influence efficiency?

Not comparable, due to different training and evaluation conditions

# How can we find the best efficient transformer?

- Which modifications and overall strategies are the most efficient?
- Are these modifications worth considering over the baseline ViT?
- What other dimensions influence efficiency?

Not comparable, due to different training and evaluation conditions

1. Train models from scratch
2. Analyze using the framework of the Pareto front

# For comparability, we train every model from scratch.

- Based on DeiT III [1], an update to the popular pipeline from DeiT [2]
- Contains only standard CV elements

**1** Pretrain
- ImageNet-21k
- 90 epochs
- Learning rate 0.003 with cosine decay

**2** Finetune
- ImageNet-1k
- 50 epochs
- Learning rate 0.0003 with cosine decay

| Model | Original DeiT | Original Accuracy | Ours Accuracy |
|---|---|---|---|
| ViT-S (DeiT) | ✓ | 79.8 | **82.54** |
| ViT-S (DeiT III) | | 82.6 | 82.54 |
| XCiT-S | ✓ | 82.0 | **83.65** |
| Swin-S | ✓ | 83.0 | **84.87** |
| SwinV2-Ti | | 81.7 | **83.09** |
| Wave-ViT-S | | 82.7 | **83.61** |
| Poly-SA-ViT-S | | 71.48 | **78.34** |
| SLAB-S | ✓ | **80.0** | 78.70 |
| EfficientFormer-V2-S0 | | **75.7**$^{D}$ | 71.53 |
| CvT-13 | | **83.3**↑ | 82.35 |
| CoaT-Ti | ✓ | 78.37 | 78.42 |
| EfficientViT-B2 | | **82.7**↑ | 81.52 |
| NextViT-S | | 82.5 | **83.92** |
| ResT-S | ✓ | 79.6 | **79.92** |
| FocalNet-S | | 83.4 | **84.91** |
| SwiftFormer-S | | **78.5**$^{D}$ | 76.41 |
| FastViT-S12 | ✓ | **79.8**↑ | 78.77 |
| EfficientMod-S | ✓ | **81.0** | 80.21 |
| GFNet-S | | 80.0 | **81.33** |
| EViT | ✓ | 79.4 | **82.29** |
| DynamicViT-S | | **83.0**$^{D}$ | 81.09 |
| EViT Fuse | ✓ | 79.5 | **81.96** |
| ToMe-ViT-S | ✓ | 79.42 | **82.11** |
| TokenLearner-ViT-8 | | 77.87↓ | **80.66** |
| STViT-Swin-Ti | ✓ | 80.8 | **82.22** |
| CaiT-S24 | ✓ | 82.7 | **84.91** |

[1] H. Touvron, M. Cord, H. Jégou. "DeiT III: Revenge of the ViT". ECCV 2022.
[2] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou "Training data-efficient image transformers & distillation through attention". PMLR 2021.

# For comparability, we train every model from scratch.

- Based on DeiT III [1], an update to the popular pipeline from DeiT [2]
- Contains only standard CV elements

**1** Pretrain
- ImageNet-21k
- 90 epochs
- Learning rate 0.003 with cosine decay

**2** Finetune
- ImageNet-1k
- 50 epochs
- Learning rate 0.0003 with cosine decay

- 13 out of 26 models are based on DeiT
- + 0.85% on average
- Up to +6.86% for Poly-SA

| Model | Original DeiT | Original Accuracy | Ours Accuracy |
|---|---|---|---|
| ViT-S (DeiT) | ✓ | 79.8 | **82.54** |
| ViT-S (DeiT III) | | 82.6 | 82.54 |
| XCiT-S | ✓ | 82.0 | **83.65** |
| Swin-S | ✓ | 83.0 | **84.87** |
| SwinV2-Ti | | 81.7 | **83.09** |
| Wave-ViT-S | | 82.7 | **83.61** |
| Poly-SA-ViT-S | | 71.48 | **78.34** |
| SLAB-S | ✓ | **80.0** | 78.70 |
| EfficientFormer-V2-S0 | | **75.7**$^D$ | 71.53 |
| CvT-13 | | **83.3**↑ | 82.35 |
| CoaT-Ti | ✓ | 78.37 | 78.42 |
| EfficientViT-B2 | | **82.7**↑ | 81.52 |
| NextViT-S | | 82.5 | **83.92** |
| ResT-S | ✓ | 79.6 | **79.92** |
| FocalNet-S | | 83.4 | **84.91** |
| SwiftFormer-S | | **78.5**$^D$ | 76.41 |
| FastViT-S12 | ✓ | **79.8**↑ | 78.77 |
| EfficientMod-S | ✓ | **81.0** | 80.21 |
| GFNet-S | | 80.0 | **81.33** |
| EViT | ✓ | 79.4 | **82.29** |
| DynamicViT-S | | **83.0**$^D$ | 81.09 |
| EViT Fuse | ✓ | 79.5 | **81.96** |
| ToMe-ViT-S | ✓ | 79.42 | **82.11** |
| TokenLearner-ViT-8 | | 77.87↓ | **80.66** |
| STViT-Swin-Ti | ✓ | 80.8 | **82.22** |
| CaiT-S24 | ✓ | 82.7 | **84.91** |

[1] H. Touvron, M. Cord, H. Jégou. "DeiT III: Revenge of the ViT". ECCV 2022.
[2] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou "Training data-efficient image transformers & distillation through attention". PMLR 2021.

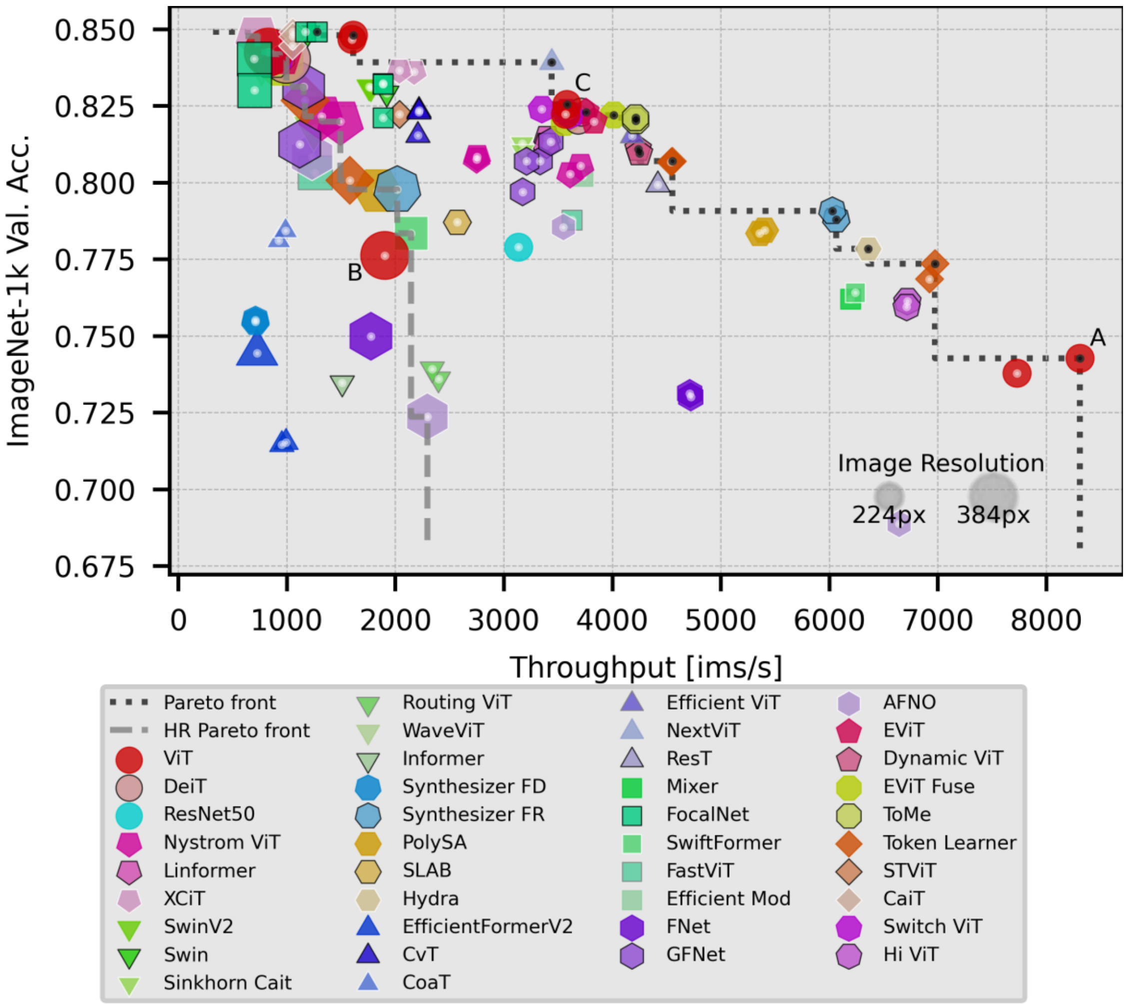
SUSTAINABLE EMBEDDED AI

WACV 2025 TUCSON, ARIZONA • FEB 28 - MAR 4

dfki ai RPTU

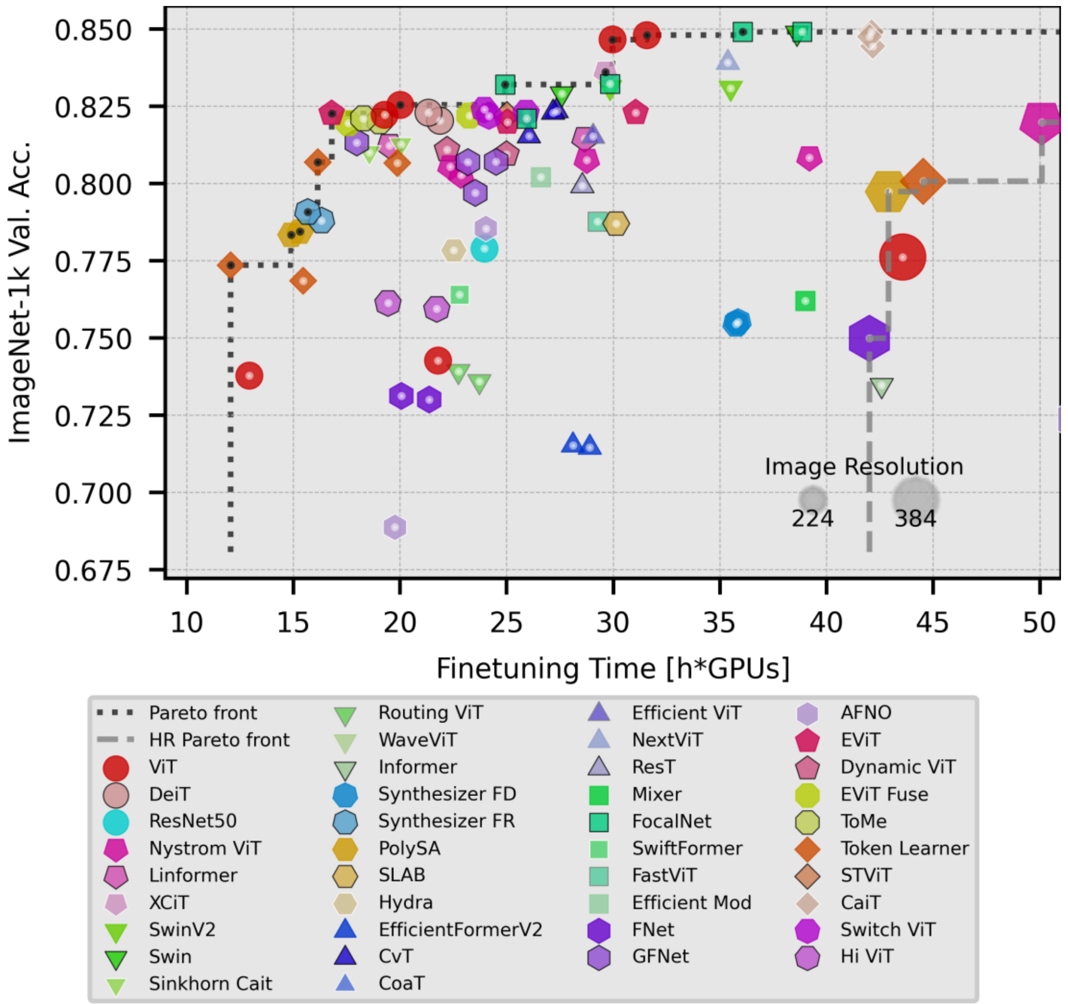
# The baseline ViT model is still Pareto optimal in terms of speed.

1. ViT is still Pareto-optimal
2. Scaling up the model size is more efficient than scaling up the image resolution
   - Short sequences for image classification
3. Sequence reduction is a way to speed up without losing too much accuracy



# The baseline ViT model is still Pareto optimal in terms of speed.

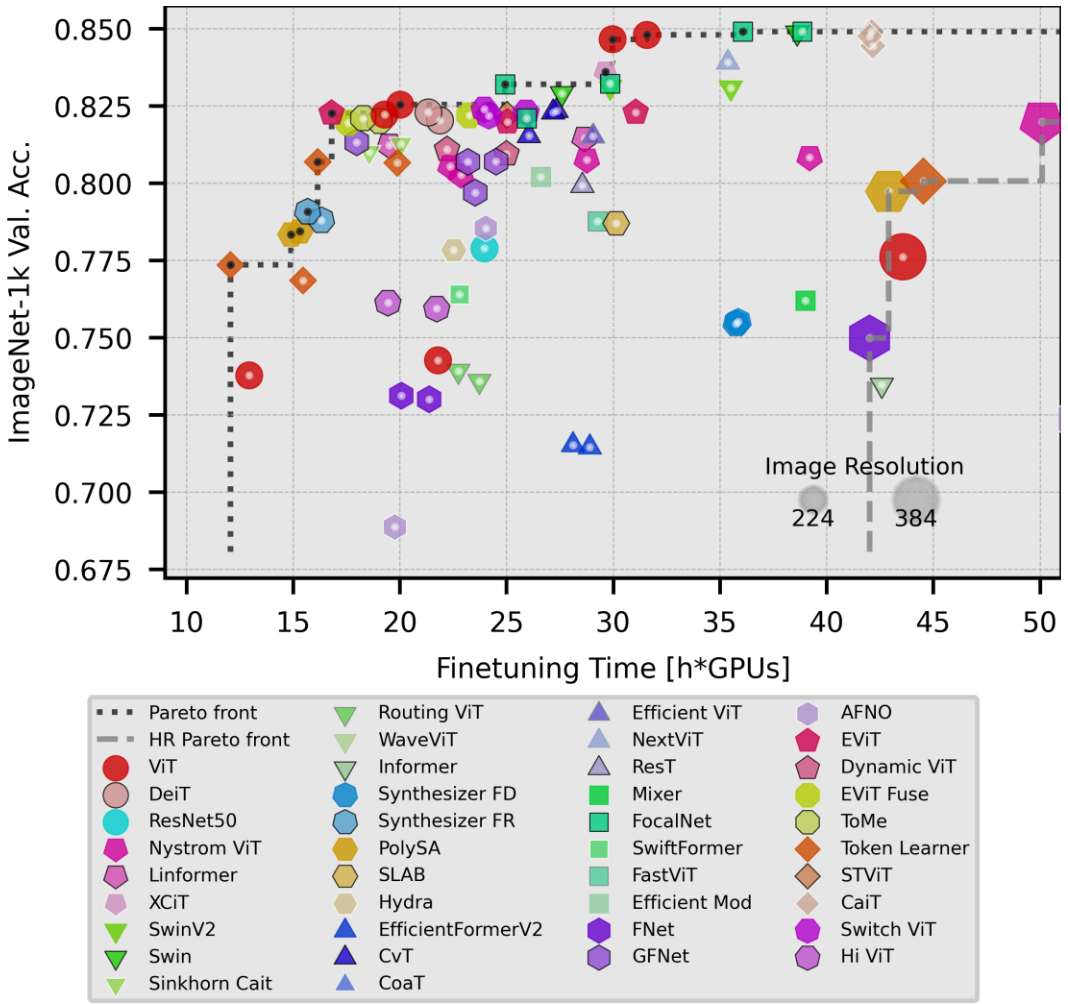
1. ViT is still Pareto-optimal
2. Scaling up the model size is more efficient than scaling up the image resolution
   - Short sequences for image classification
3. Sequence reduction is a way to speed up without losing too much accuracy



# The baseline ViT model is still Pareto optimal in terms of speed.

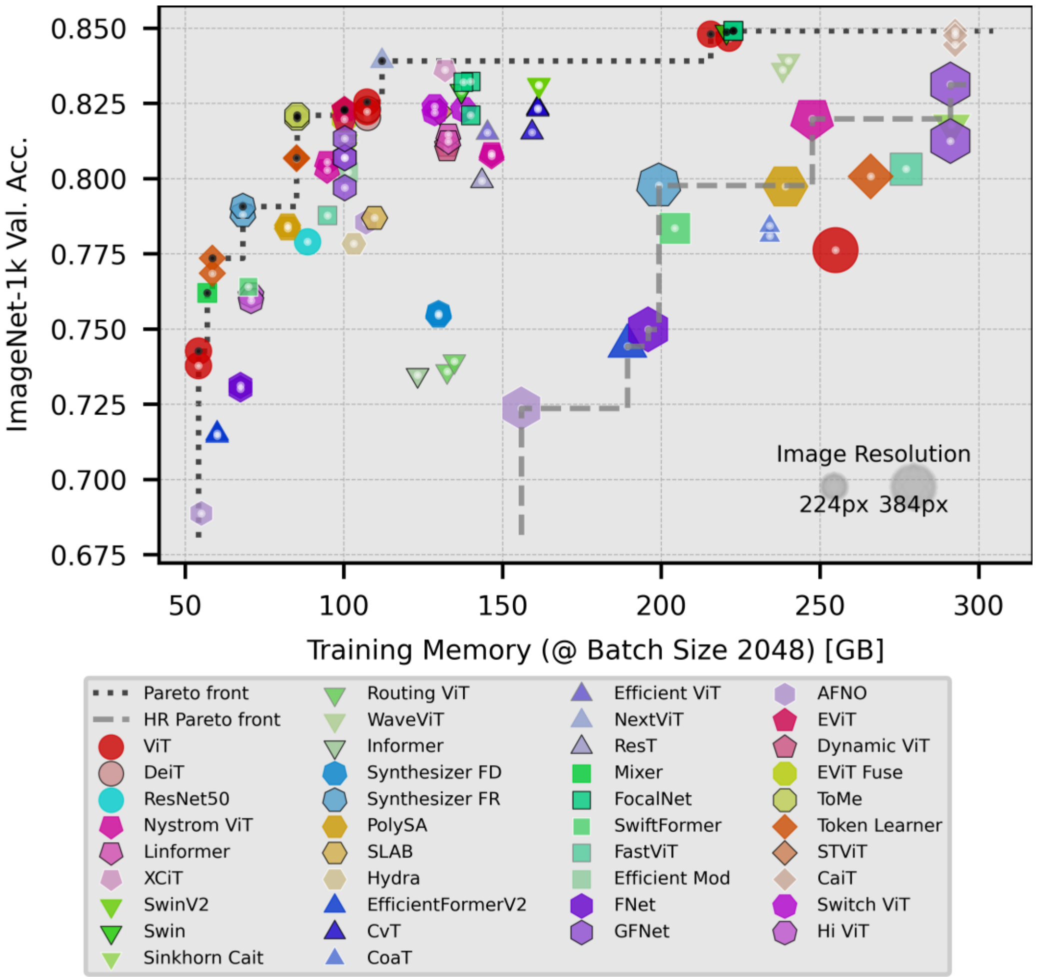
1. ViT is still Pareto-optimal
2. Scaling up the model size is more efficient than scaling up the image resolution
   - Short sequences for image classification
3. Sequence reduction is a way to speed up without losing too much accuracy

- Pareto front replicates on other datasets with a spearman correlation of >0.71
- And on other devices with a spearman correlation of >0.75



# Convolution-based models are very inference-memory efficient.

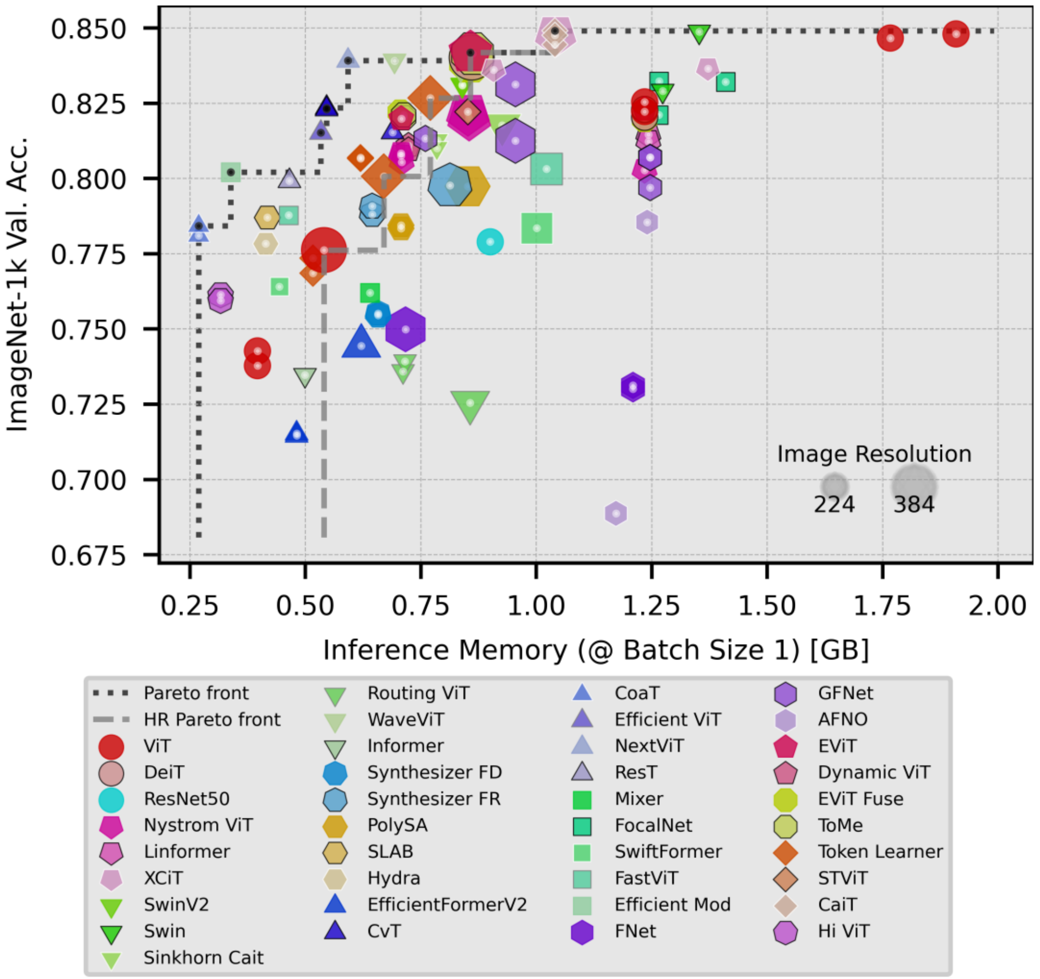
**Training Memory:**
Very similar to speed



# Convolution-based models are very inference-memory efficient.

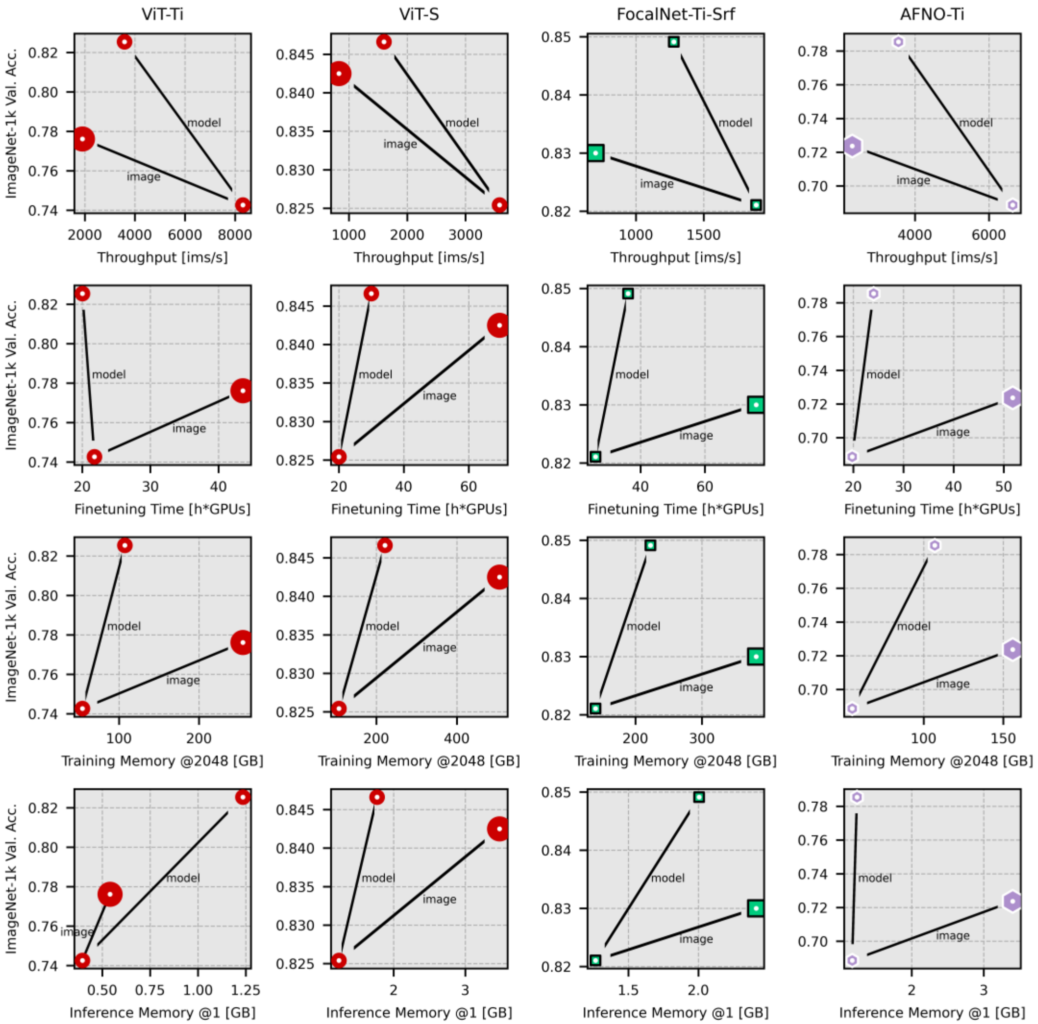
**Training Memory:**
Very similar to speed

**Inference Memory:**
Different from the other metrics
1. ViT is not Pareto optimal
2. Models incorporating convolutions excel in this metric
3. EViT @ 384 is the only Pareto optimal model using high-resolution images



# Use larger models, not larger images.

- Using high resolution images (384 x 384 px) is not Pareto optimal
- Tradeoff of scaling up the model size is better than for scaling up the image resolution

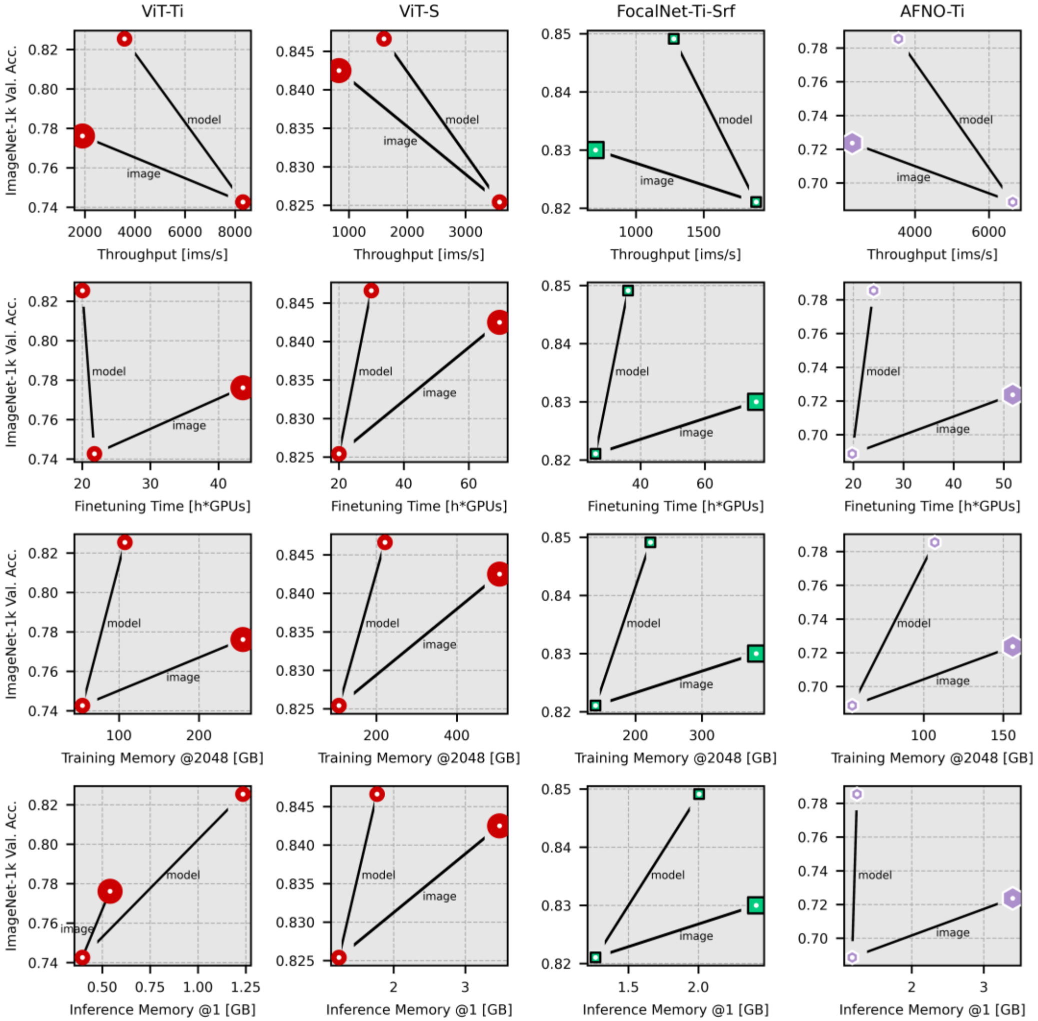


# Use larger models, not larger images.

- Using high resolution images (384 x 384 px) is not Pareto optimal
- Tradeoff of scaling up the model size is better than for scaling up the image resolution

- Using a larger model with 224px images is 2 to 3 times faster than a smaller model with 384px images

  Also uses 2 to 3 times less training memory

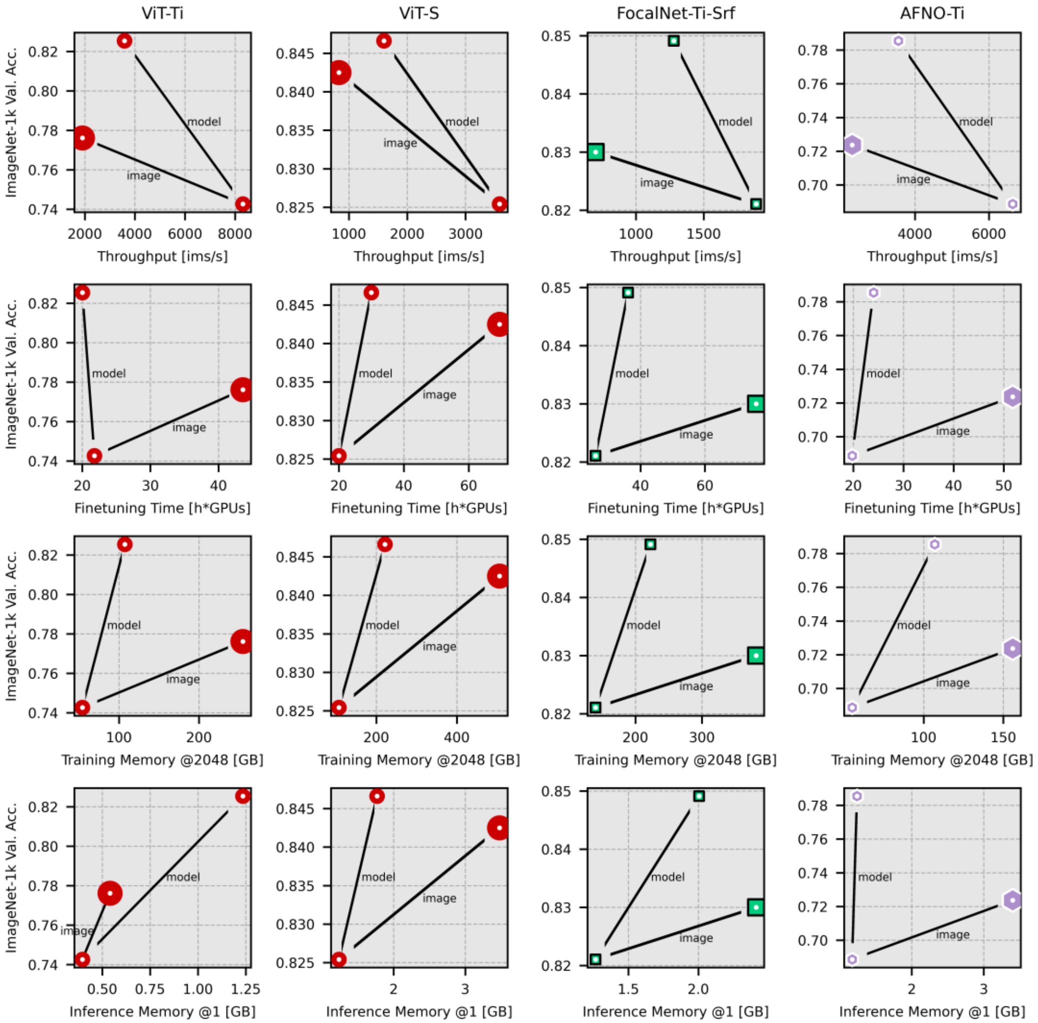


# Use larger models, not larger images.

- Using high resolution images (384 x 384 px) is not Pareto optimal
- Tradeoff of scaling up the model size is better than for scaling up the image resolution

- Using a larger model with 224px images is 2 to 3 times faster than a smaller model with 384px images
  Also uses 2 to 3 times less training memory

**Interpretation**:
In classification, the goal is to synthesize the information down to 1-d. Therefore, fine-grained information is not needed.



# Thanks & Goodbye

WTF Benchmark – Tobias Nauen – WACV 2025

Questions?

Feel free to reach out!

Tobias Nauen

tobias_christian.nauen@dfki.de

PDF

Code

Analysis



This work was supported by the Carl-Zeiss Foundation under the Sustainable Embedded AI project (P2021-02-009)