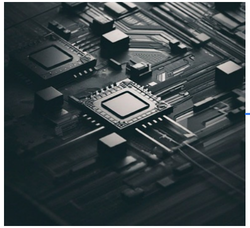
MatFormer: Nested Transformer for Elastic Inference

Prateek Jain
Google Research India

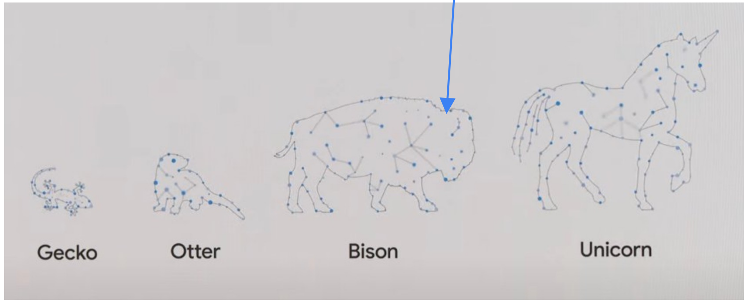
Devvrit, Sneha Kudugunta, Aditya Kusupati,
Tim Dettmers, Hannaneh Hajishirzi, Yulia Tsvetkov, Kaifeng Chen, Inderjit Dhillon,
Sham Kakade, Ali Farhadi

Large Models: Deployment Story



Deployment Constraints
(RAM, Latency, QPS…)

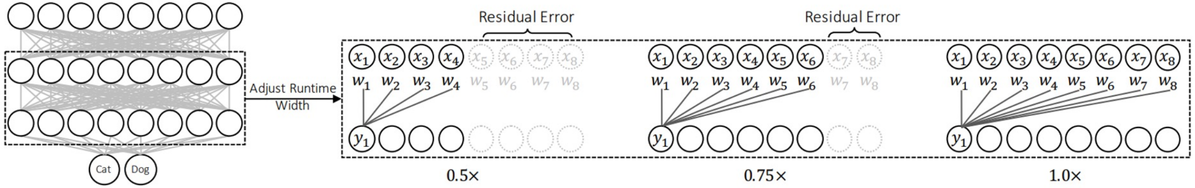
- Typically only a few models to choose from
  - Might have to select say Llama 13B even if capacity for Llama 40B
- Distillation/pruning requires additional training



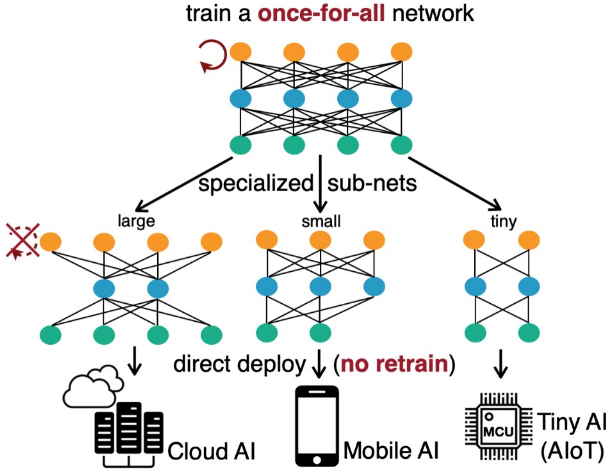
PALM 2

Goal: *design a "universal" model from which hundreds of accurate models can be extracted*

Existing Solutions towards MatFormer



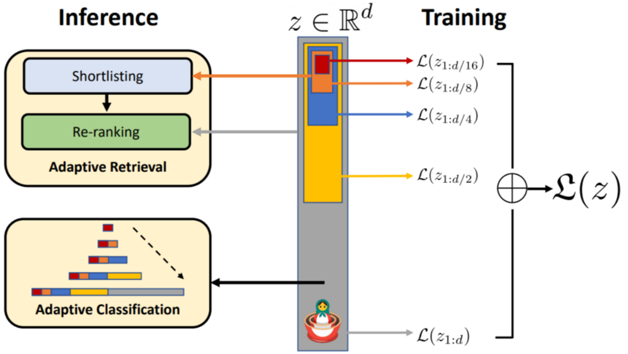
(Universal) Slimmable Networks (Yu & Huang ICLR 2019; ICCV 2019)



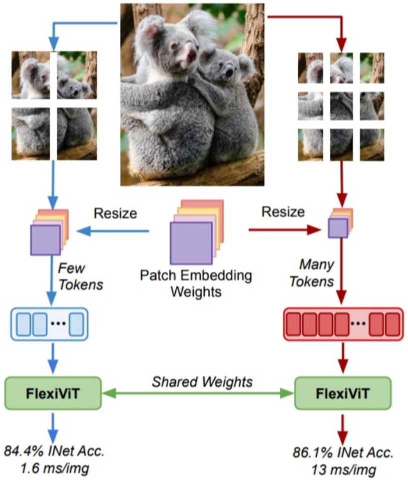
Once-for-All (Cai et al., ICLR 2020)

- Primarily focused on CNNs
- Training routines w/
    - Modifications to batchnorm
    - Sampled submodels
    - Distillation from largest model
- Longer/costlier training routines

Existing Solutions towards MatFormer



Matryoshka Representation Learning (Kusupati et al., NeurIPS 2022)



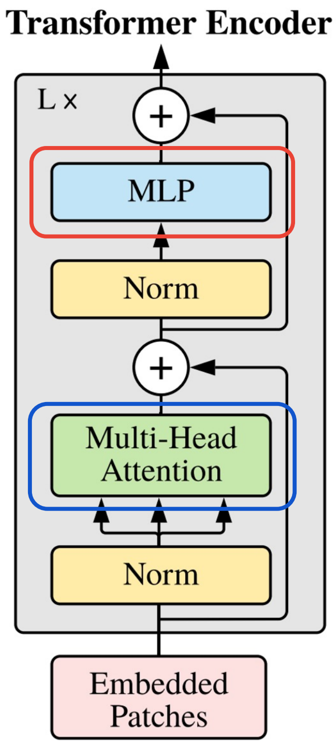
FlexiViT (Beyer et al., CVPR 2023)

Flexibility in output and input space respectively

MatFormer: Nested Substructure

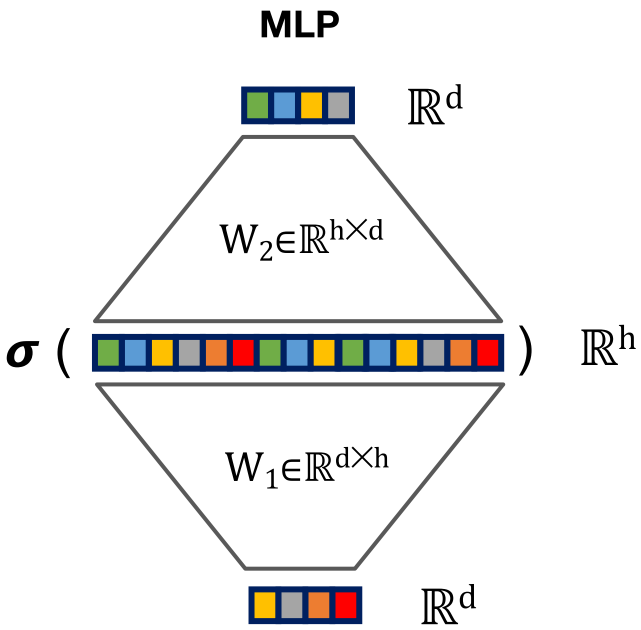
- Works for Transformers – without modifications to fundamental blocks
- Joint training: No subsampling or distillation
- Significantly cheaper training cost that equivalent methods while being more accurate
- 4 granularities sufficient to span a wide range of constraints.

Transformer



MLP: **~80%** of the cost

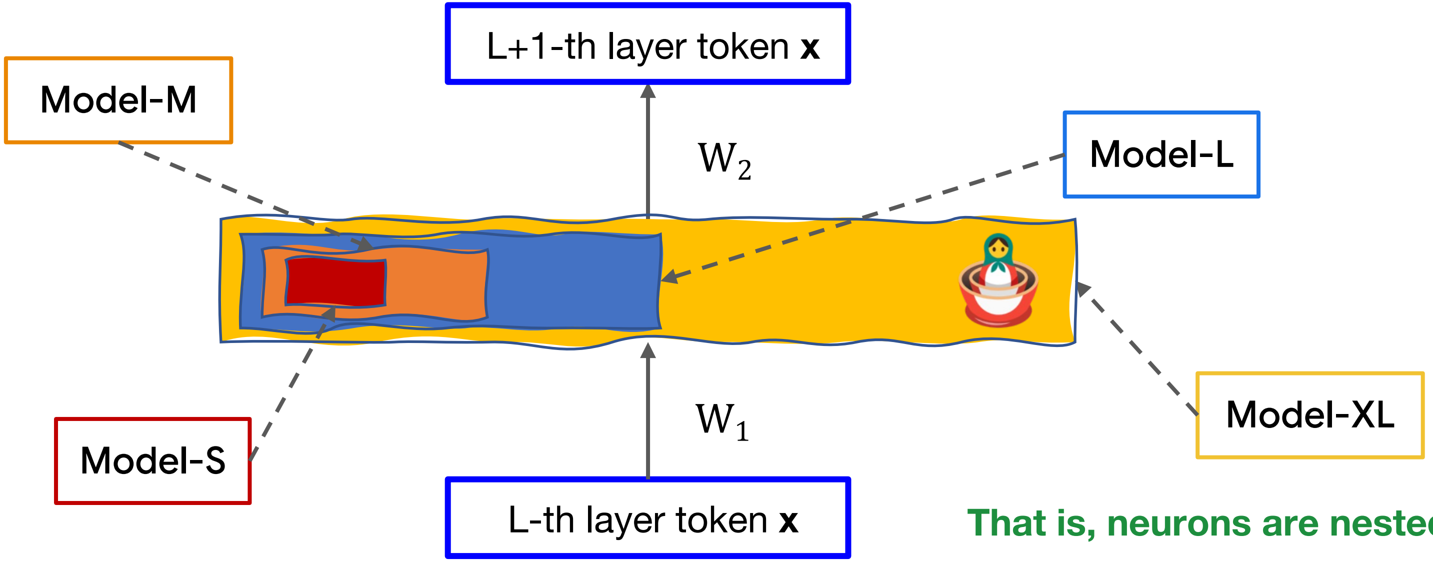
Attention: **~20%** of the cost



h = **4*d** to **12*d** typically

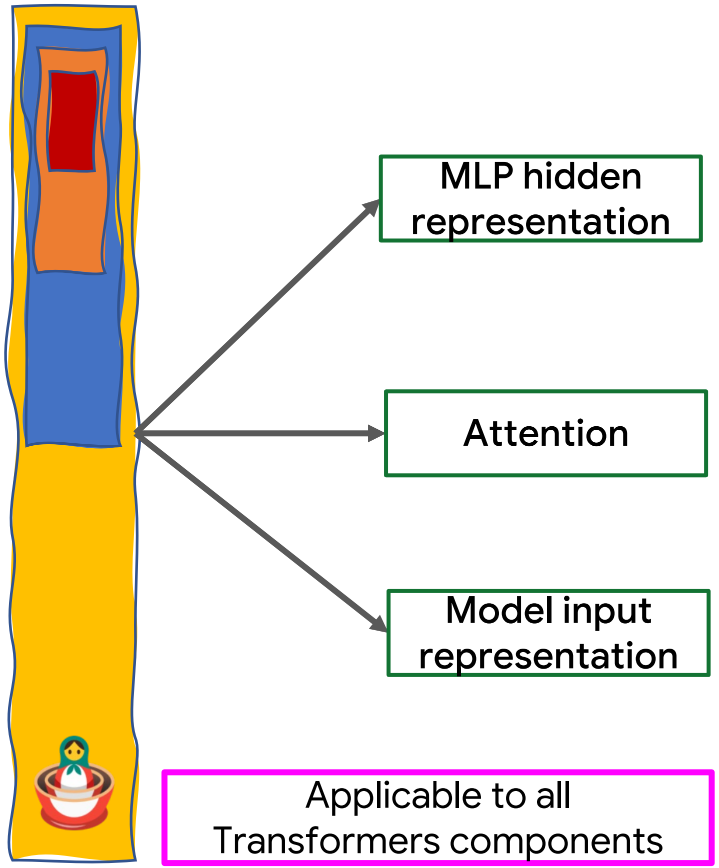
MatFormer: Matryoshka Transformer

- **MatFormer** builds upon [MRL]
- Apply MRL to MLP layer in each transformer block



That is, neurons are nested in each other.

MatFormer: Generality & Training

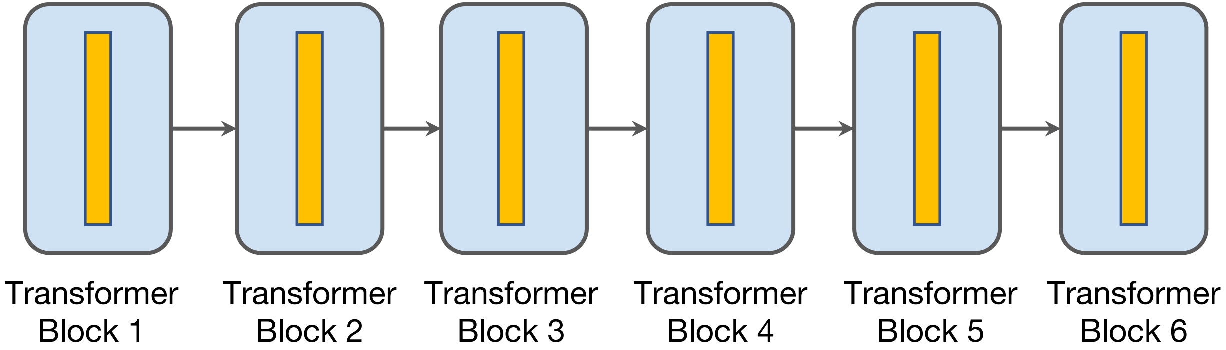


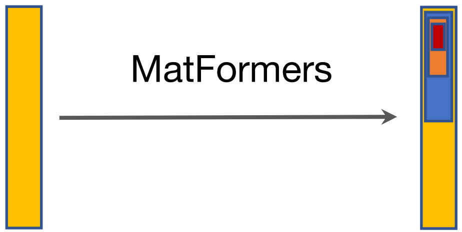
Recipe:

- Pick XL model architecture
- Pick **G** granularities for nesting eg., **G = 4**
- Jointly optimize **G** shared models akin to MRL
- Matformer train cost **<** **total cost of training each granularity from scratch**
- MatFormer can also be induced w/ Fine-tuning

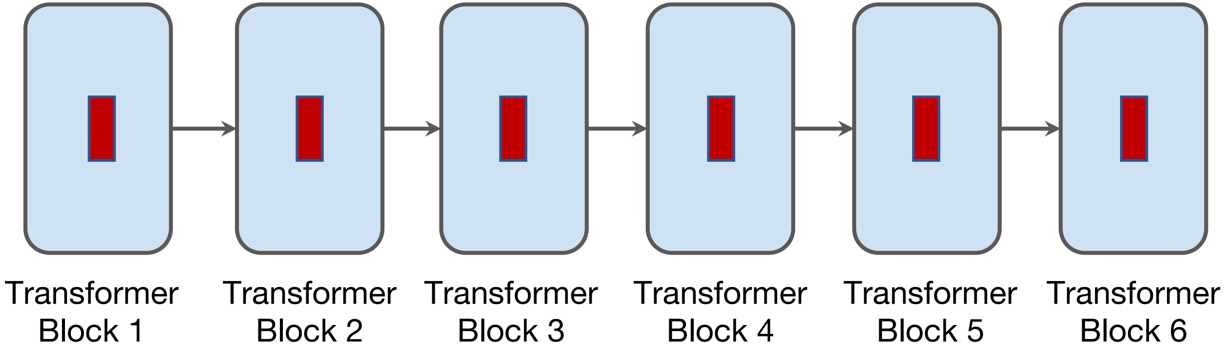
Can generate **1000s** of models not just **4**

Mix'n'Match & Routing on MatFormer





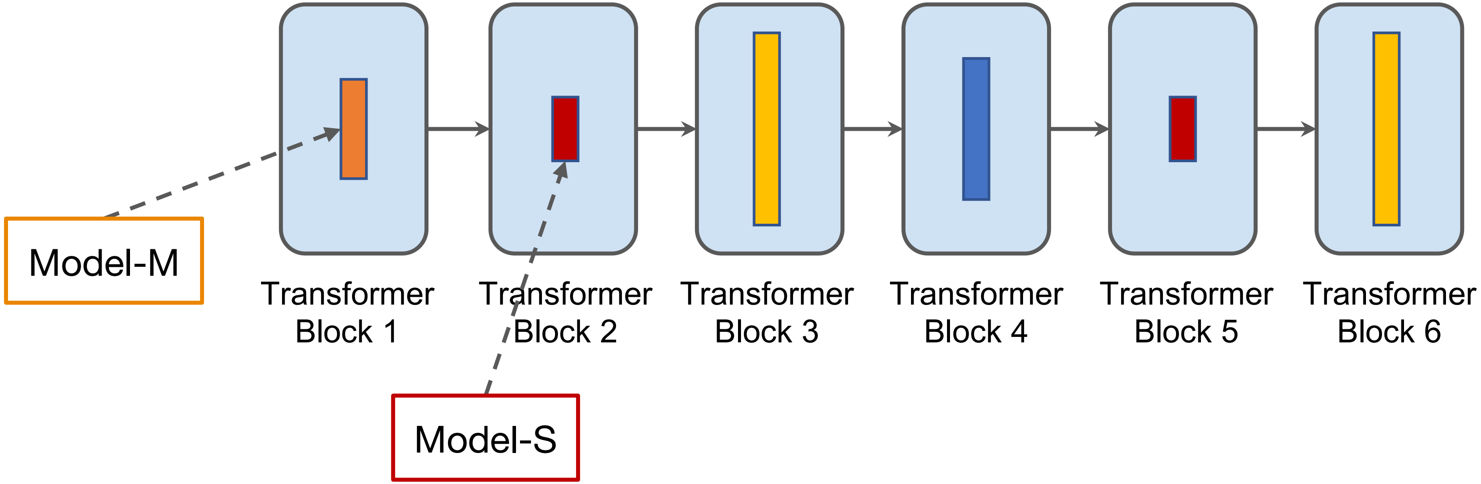
Mix'n'Match & Routing on MatFormer



This gives only say 4 models.

So where do we get 1000s of models???

Mix'n'Match & Routing on MatFormer



- **Mix’n’Match:** 100s (combinatorial) of *static (on-demand)* models for all accuracy-compute
- **Routing:** Token based routing akin to MoE to realize *dynamic* computation

MatLM: MatFormers for Language Modeling

- Standard setting from Lamda (Thoppilan et al.)
- **G = 4** granularities – change the MLP hidden dims!
  - **XL** – hidden_dim (hd), **L** – hd/2, **M** – hd/4, **S** – hd/8.
- Nomenclature: MatFormer-XL, MatFormer-L, MatFormer-M, MatFormer-S
  - Independently Trained Models: Baseline-XL, Baseline-L, Baseline-M, Baseline-S
- *7 different "XL" model scales:* from 78M up to 2.6B parameters.
  - 78M, 180M, 310M, 463M, 850M, 1.3B, 2.6B

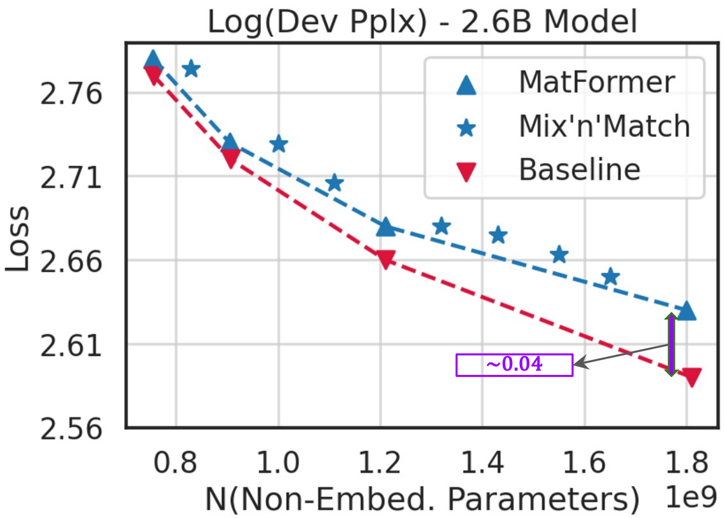
MatLM: Key Findings

- Little to **no loss in test pplx** and **GPT3 1-shot downstream evals**.
  - For each granularity, i.e., accuracy(MatFormer-Z) ~ accuracy(Baseline-Z)
  - Z ∈ [XL, L, M, S]

- Able to read models for *free* using Mix'n'Match
  - Mix'n'Match interpolates well between the 4 granularities

- Side effects: consistency with large model, gains over Speculative Decoding

Language Modeling with 2.6B model: Mix’n’Match



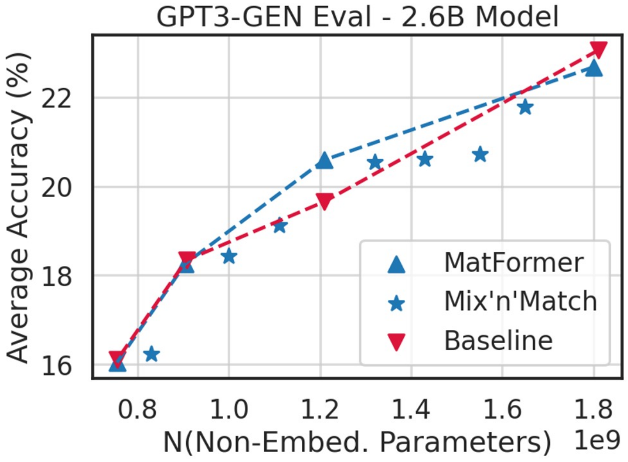
Trained for: 160B tokens

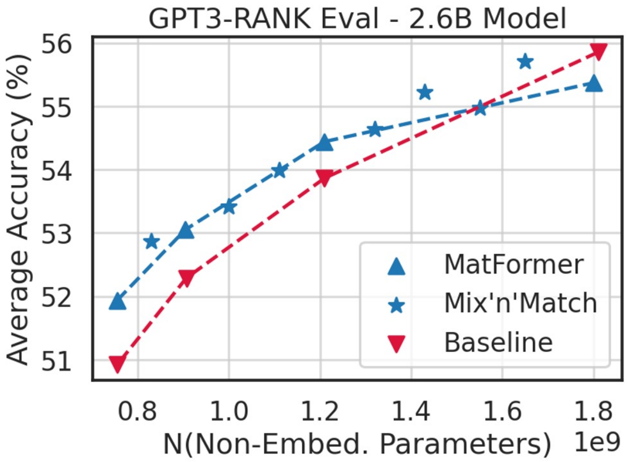
Decoder-only model w/ same hparameters for baseline & MatFormer

- XL model size: 2.6B
  - We read-off L, M, S models for matformer
  - Baselines trained from scratch for all granularities
- Log pplx within 0.04 of the baseline for 2.6B model!
  - Downstream evals almost match (see next slide)
- All the ★ are for “free” – *they were never trained for*
  - We just read them from Matformer-XL

Language Modeling with 2.6B model: Mix’n’Match

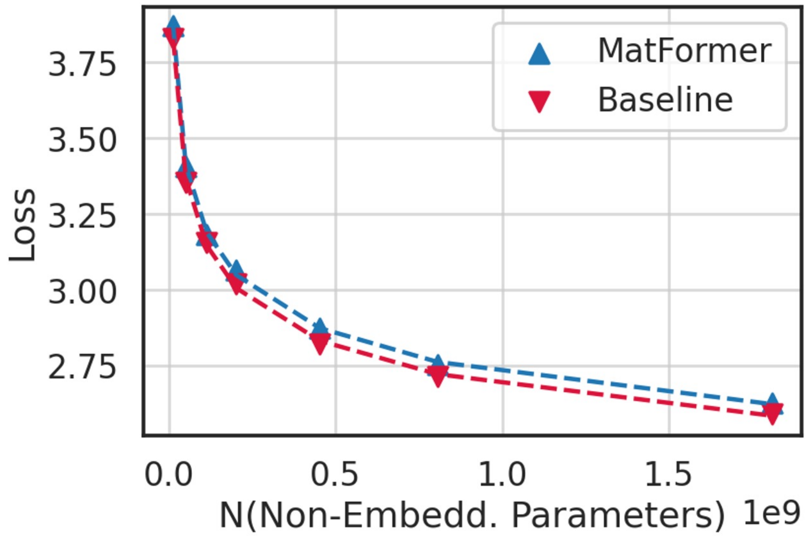
1-shot GPT-evals

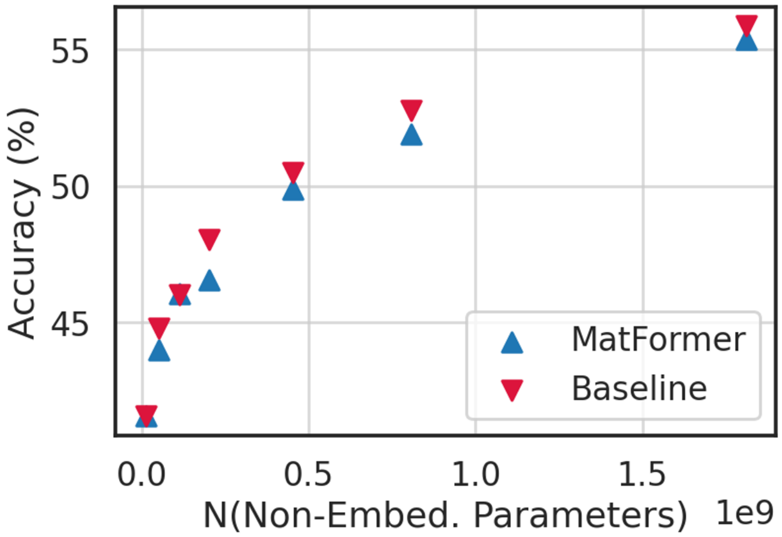




- Almost matching accuracy for MatFormer–[XL, L, M, S] models against Baselines
- We get all the intermediate models denotes by ★ for “free”
  - No extra training!
  - For GPT-3 Rank: ★ models almost lie on a line interpolating trained MatFormer models (XL, L, M, S)

Language Modeling: Scaling Plots for Universal Model (XL)





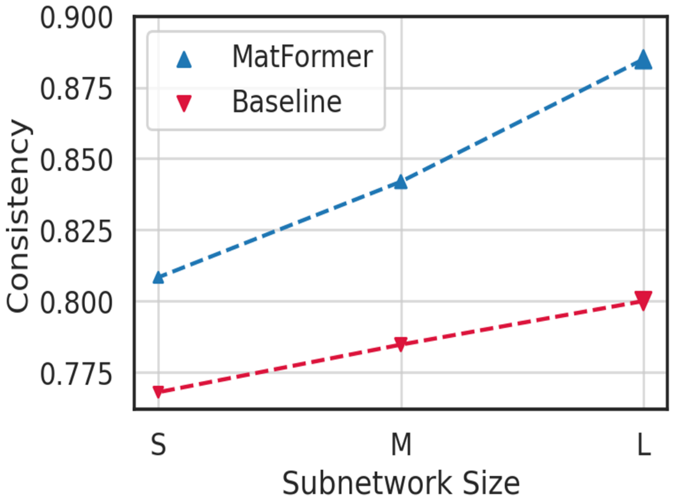
MatFormer : 23.0528 * N ^ -0.1407 + 1 / D + 1.4352

Baseline : 22.7879 * N ^ -0.1414 + 1 / D + 1.4973

Where N = N(Non-Embedding Parameters) and D = N(Training Tokens)

Above laws hold for all trained granularities: XL, L, M, S!

Language Modeling: Consistency for 2.6B XL model



***Consistency**: accuracy of smaller models (S, M, L) when output of XL model is the ground truth*

Why care about consistency?
Techniques like Speculative Decoding becomes more efficient with more consistent models

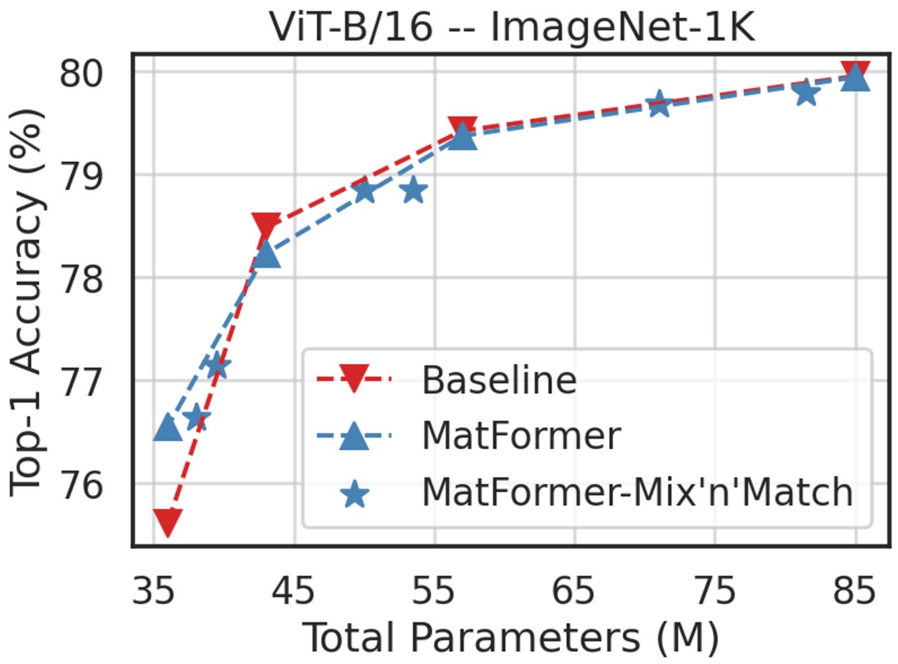
Speculative Decoding	LAMBADA	TriviaQA
Baseline	1.10×	1.08×
MatLM	1.14×	1.11×
+ shared attention cache	1.16×	1.14×

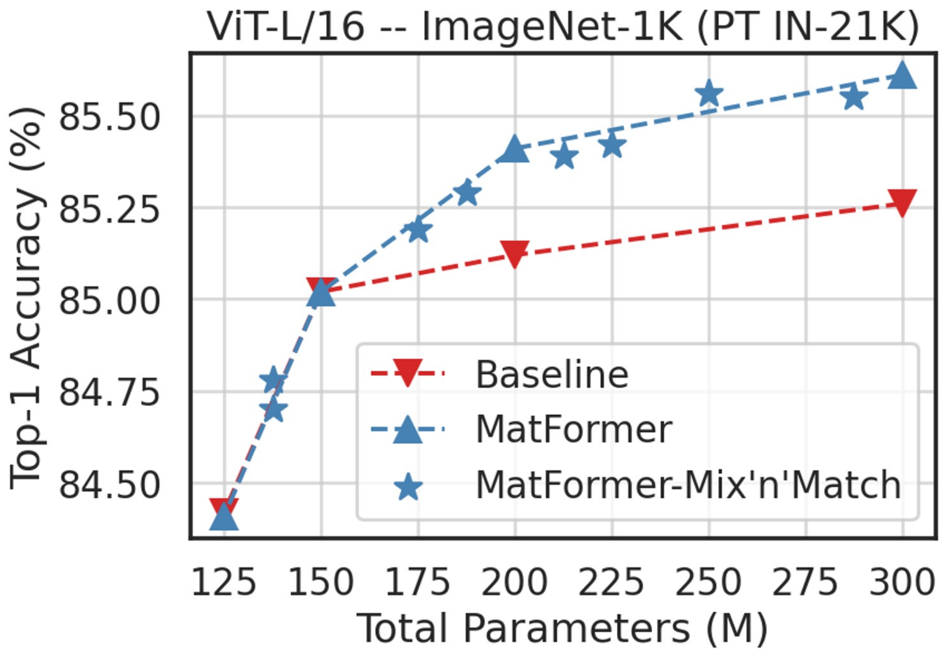
MatFormer subnetworks are significantly more consistent with the full model compared to vanilla baselines.

MatViT: MatFormer + ViT

- Generalized formulation translating to ViT
- Works for across model sizes for both pre-training and fine-tuning
- Enables accurate adaptive encoders for classification
    - Spans all of the space with Mix’n’Match (and potentially routing)
- Enables accurate adaptive query encoders for retrieval
    - Use the largest model for Index building
    - Leverage smaller query encoders during inference based on the constraints
    - This requires aligned training/distillation for baseline models to work

MatViT: Classification



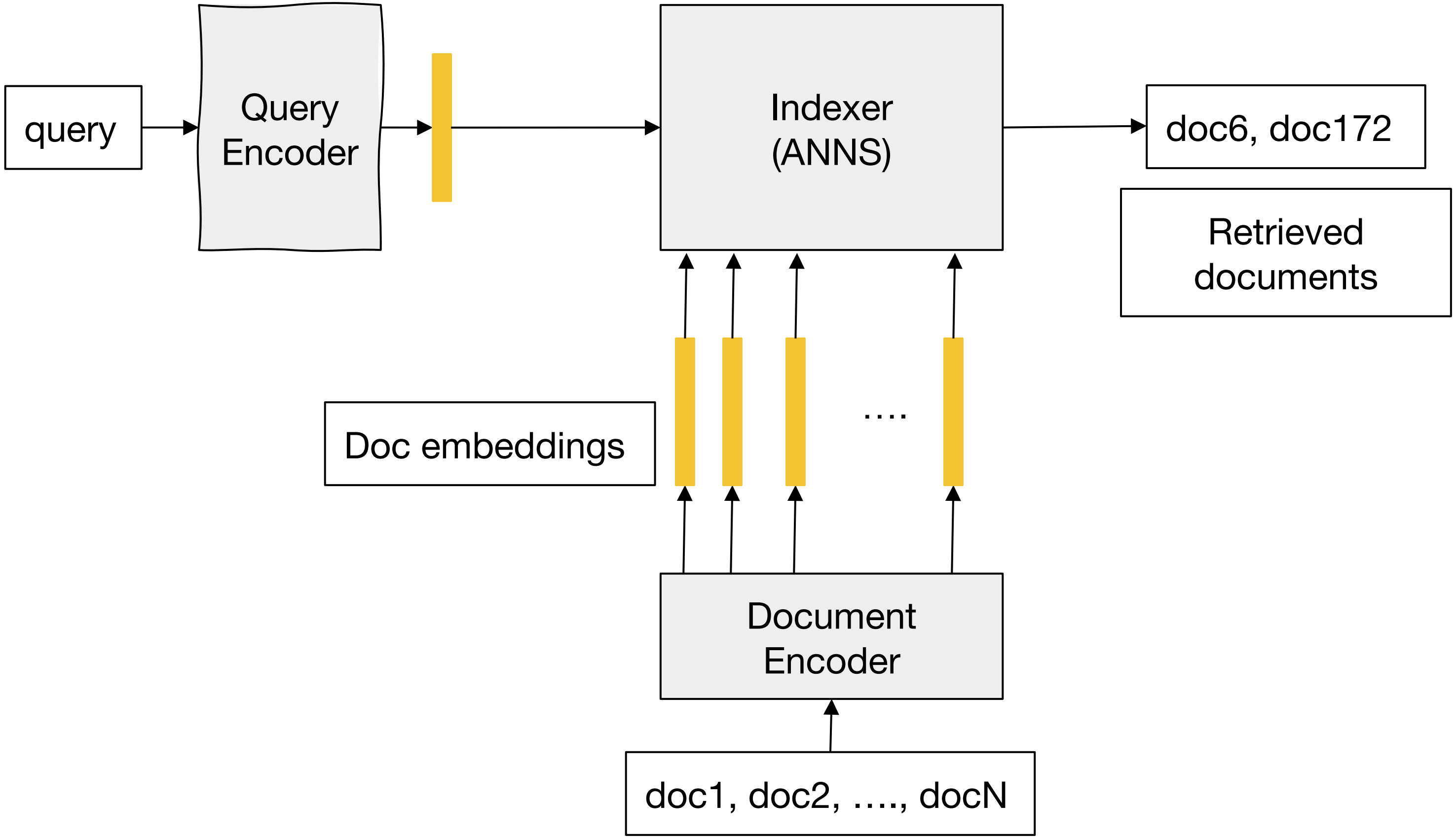


All the ★ are for “free” during inference – *they were never optimized for.*

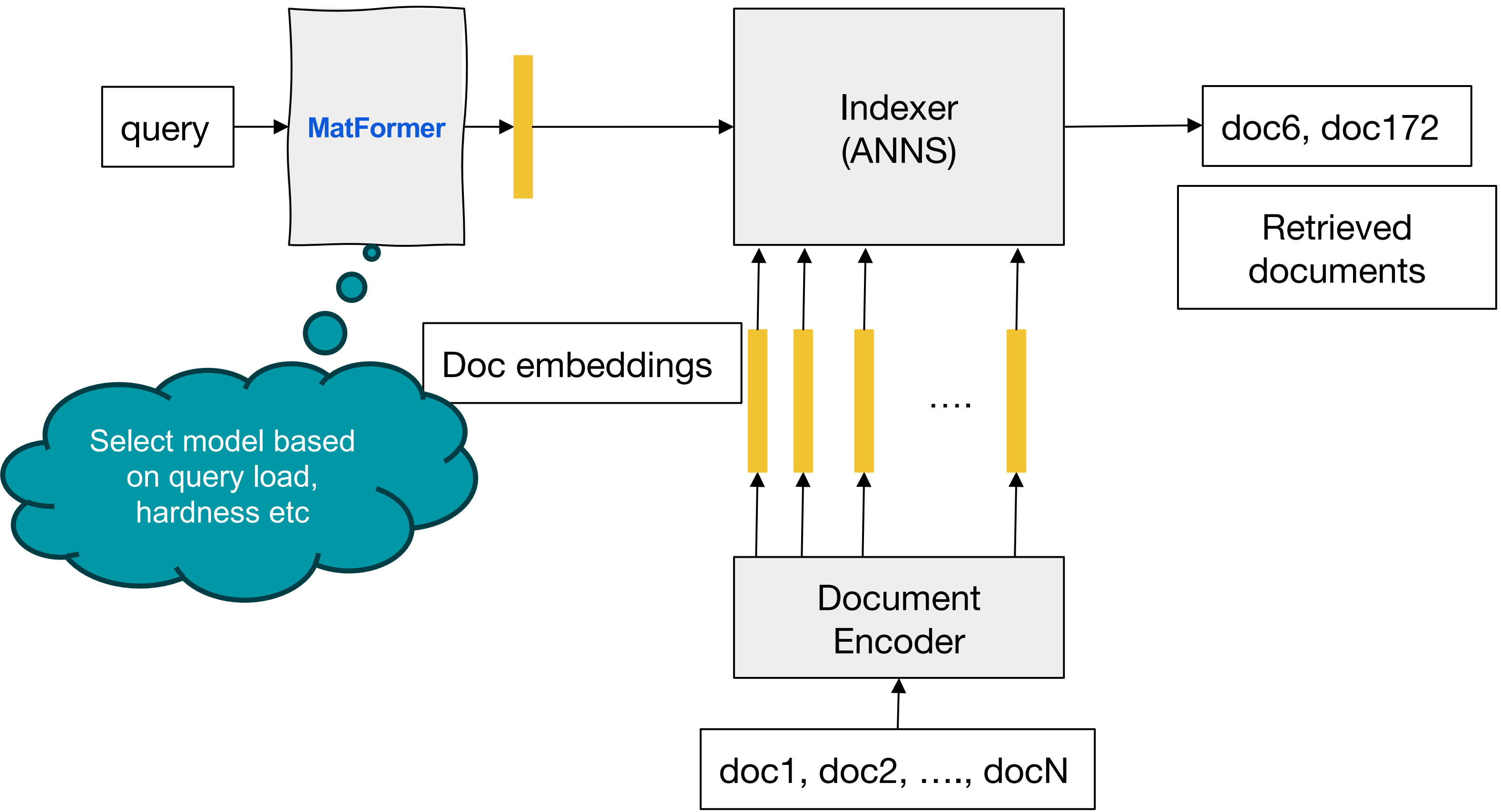
MatViT: MatFormers + ViT

- Generalized formulation translating to ViT
- Works for across model sizes for both pre-training and fine-tuning
- Enables accurate adaptive encoders for classification
  - Spans all of the space with Mix’n’Match (and potentially routing)
- Enables accurate adaptive query encoders for retrieval
  - Use the largest model for Index building
  - Leverage smaller query encoders during inference based on the constraints
  - This requires aligned training/distillation for baseline models to work

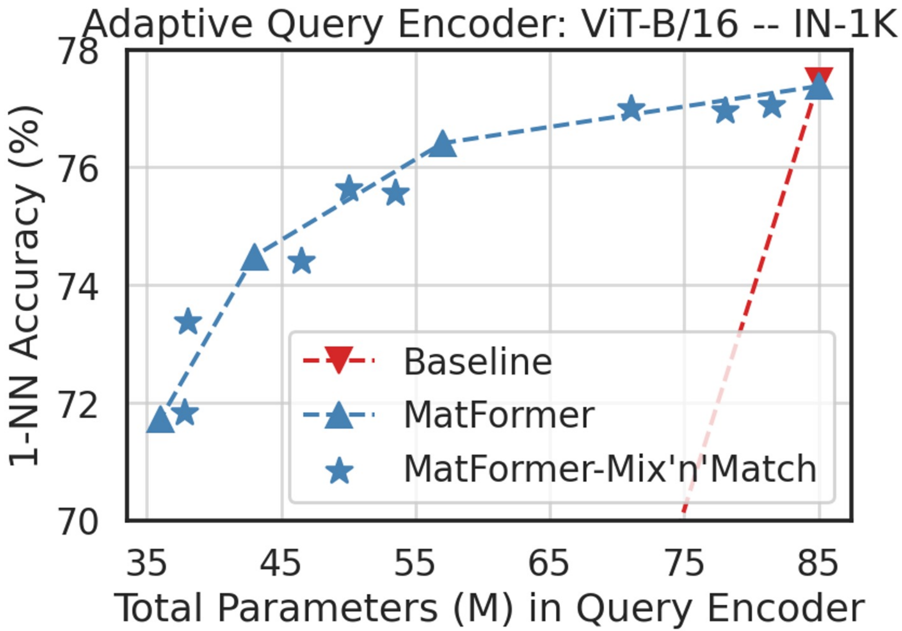
Semantic Search: Dual Encoder Models

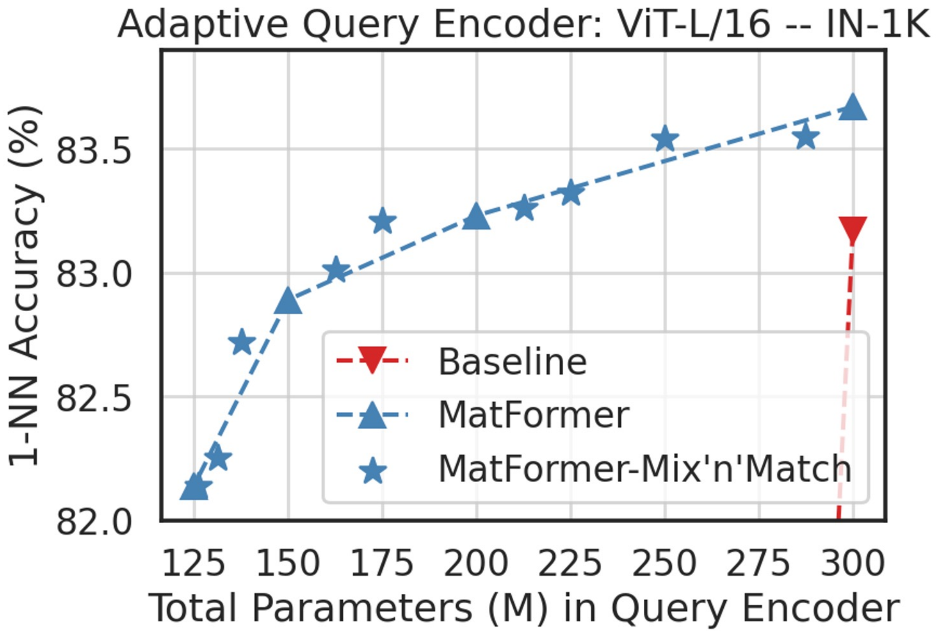


Semantic Search: Flexible Dual Encoder Model



MatViT: Adaptive Retrieval (Index built w/ largest model)





All the ★ are for “free” during inference & preserve metric space.

L/16 pretrained on IN-21K

MatFormer + ViT-B/16: Cross-consistent Retrieval

1-NN accuracy (%) with varying index and query encoder sizes from MatViT-B/16
(Baseline numbers): Rest are near Random

Index↓/Query→	36M	43M	57M	85M
36M	**72.42%** (71.44%)	74.31%	75.33%	76.26%
43M	72.30%	**74.71%** (74.90%)	75.93%	76.69%
57M	72.12%	74.71%	**76.44%** (76.58%)	77.19%
85M	71.71%	74.48%	76.40%	**77.40%** (77.46%)

Adaptive Query Encoders for Retrieval

Summary and Future Work

- Matformer: nested substructure for elastic inference
- Training: joint optimization of a few granularities
- MatLM: Matformer Language Model
  - 2.6B scale models with same pplx and evals as independently trained baselines
  - Consistency: gains over speculative decoding
- MatVIT: Matformer Vision Transformer
  - ViT-L/16 scale models, similar performance as independently trained baselines
  - Adaptive retrieval

Future Work:

- Further investigation of scaling laws
- Better training algorithms
- Practical deployment of a truly elastic system