DyTox: Transformers for Continual Learning with DYnamic TOken eXpansion (CVPR 2022)

- Young Jo Choi
- Department of Digital Analytics

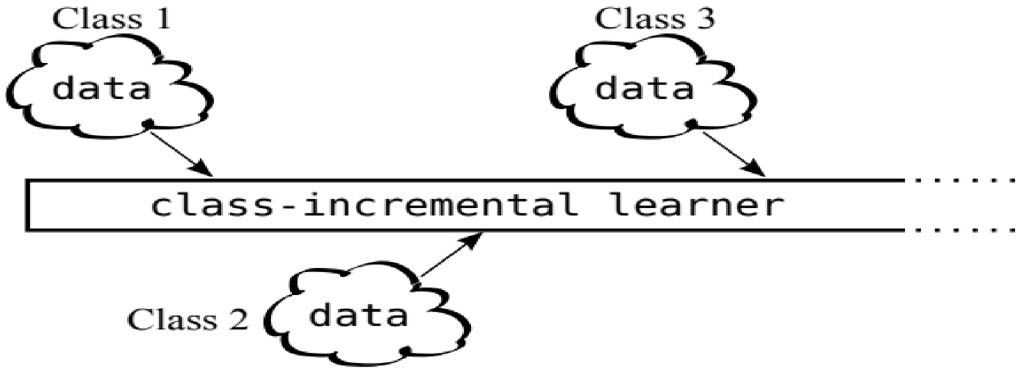
Abstract

- A recent trend in continual learning indicates that **dynamic architectures** based on an expansion of the parameters can reduce catastrophic forgetting efficiently. (ex) DER)

- However, they require complex tuning to balance the growing number of parameters and suffer from overhead as they share little information between each task's architecture.

- In this paper, the authors propose a **transformer architecture** based on a dedicated encoder/decoder framework.
  (the encoder and decoder are shared among all tasks)

Continual Learning



- With new classes of data constantly appearing, simple fine-tuning suffers from **catastrophic forgetting** data which is already learned.

- Continual learning models aim at balancing a **stability/plasticity trade-off** where old data are not forgotten (stability to changes) while learning new incoming data (plasticity to adapt).

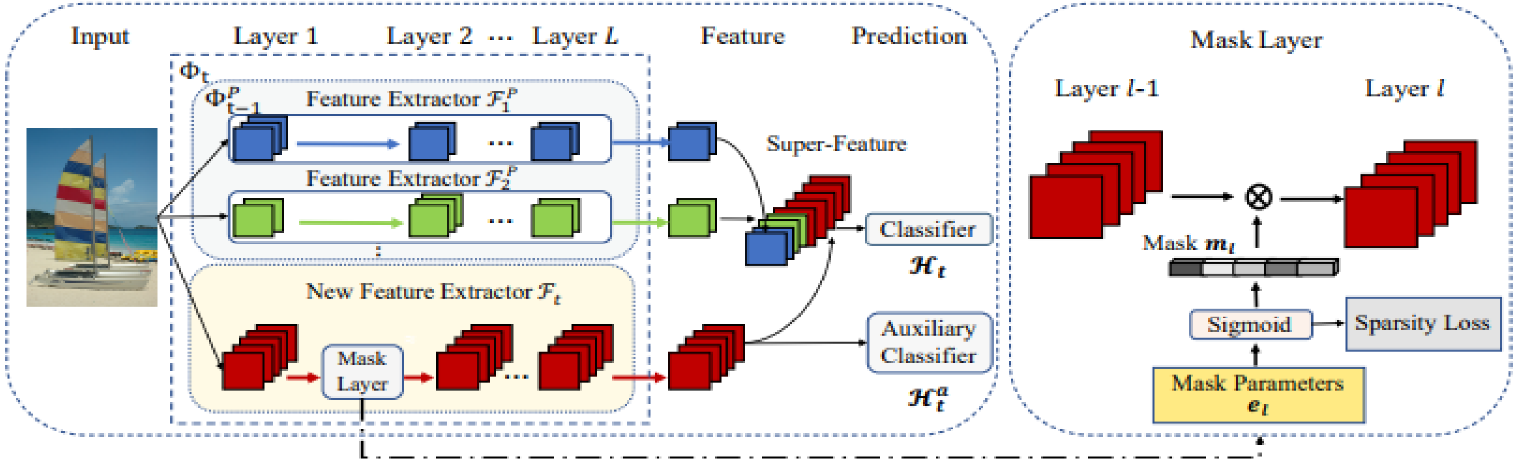
Continual Dynamic Networks

- Be able to cope with growing training distributions by expanding subnetworks for each task.

- However, in this extending method, as the tasks are added, the model size continues to increase linearly. (**memory-overhead**)

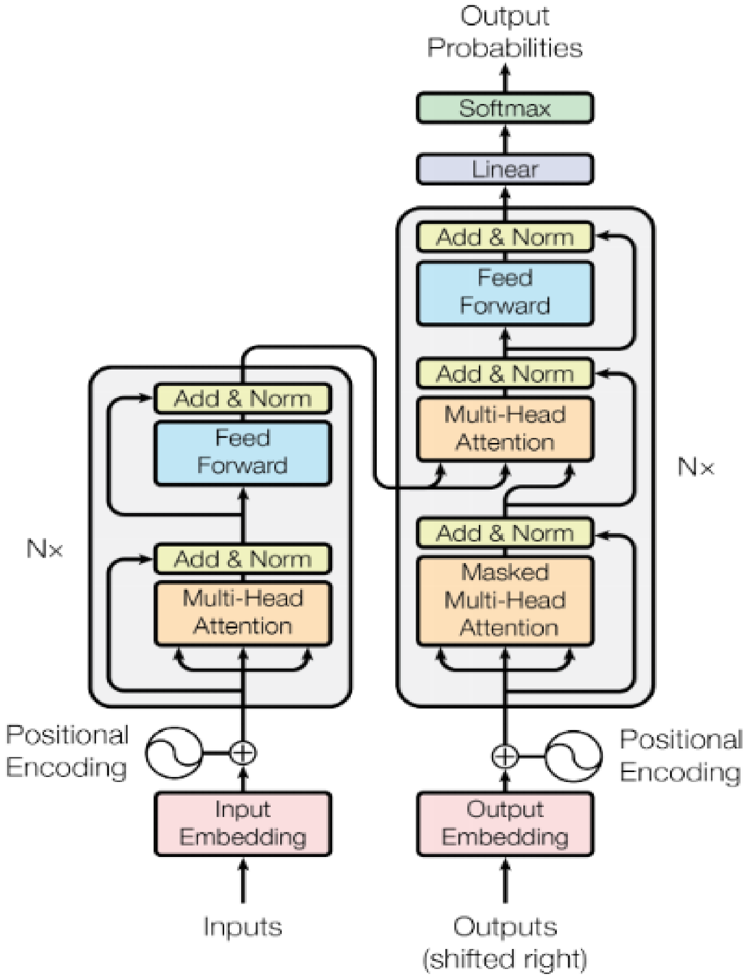
- More recently, **DER** has achieved state-of-the-art performance by concatenating the representation of an image by extending a feature extractor per task and using unified classifier to eliminate the need to use task identifiers in testing.

Continual Dynamic Networks

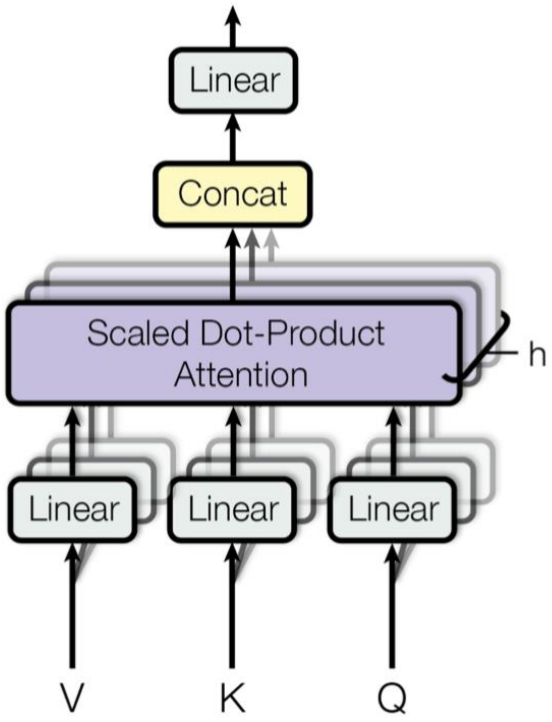


- DER uses a channel-level mask to prune to reduce the growing model size, but it is sensitive to hyperparameters, so the hyperparameters are tuned differently for each experiment.
- Ex) Different hyperparameters are used depending on whether the same dataset is learned by 10 steps or 50 steps.
- Therefore, it can be said to be **unrealistic** in true continuous learning <u>where the number of classes is unknown in advance.</u>

Transformers



Multi-Head Attention



1) Get the Q, K, and V vectors from the product of the embedding vector and the weight matrix for each input word.

2) Attention score

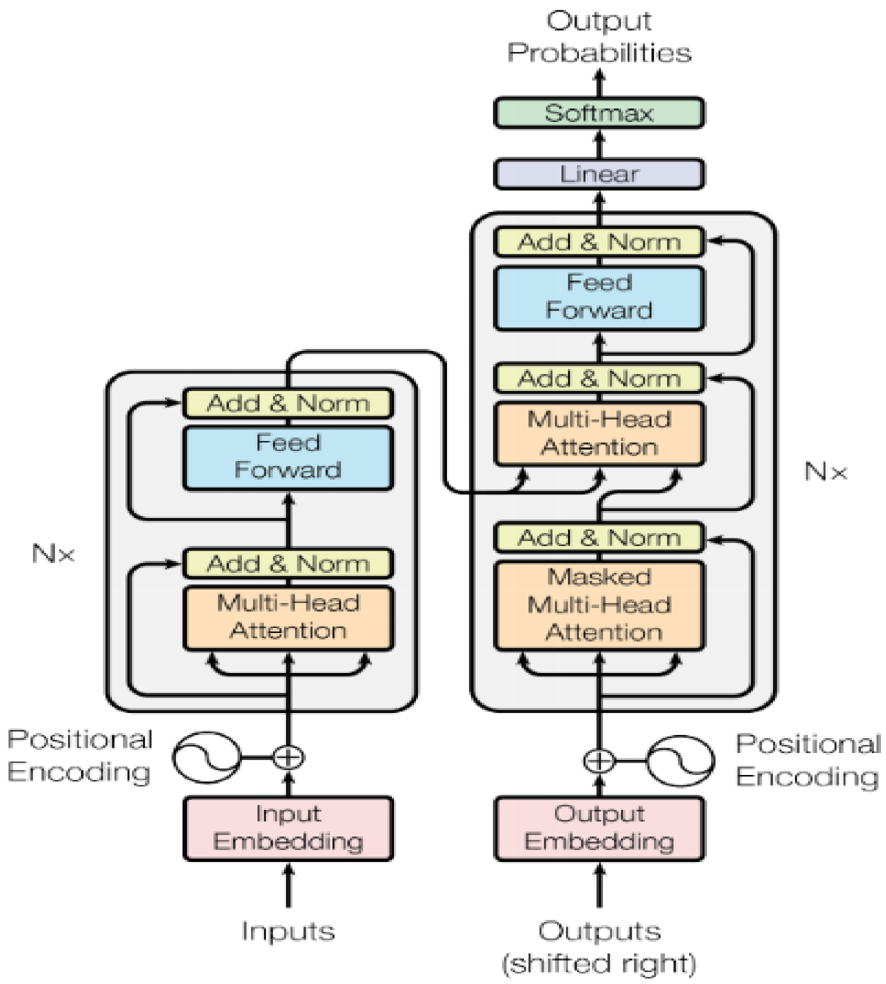
$$Attention(Q,K,V) = Softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

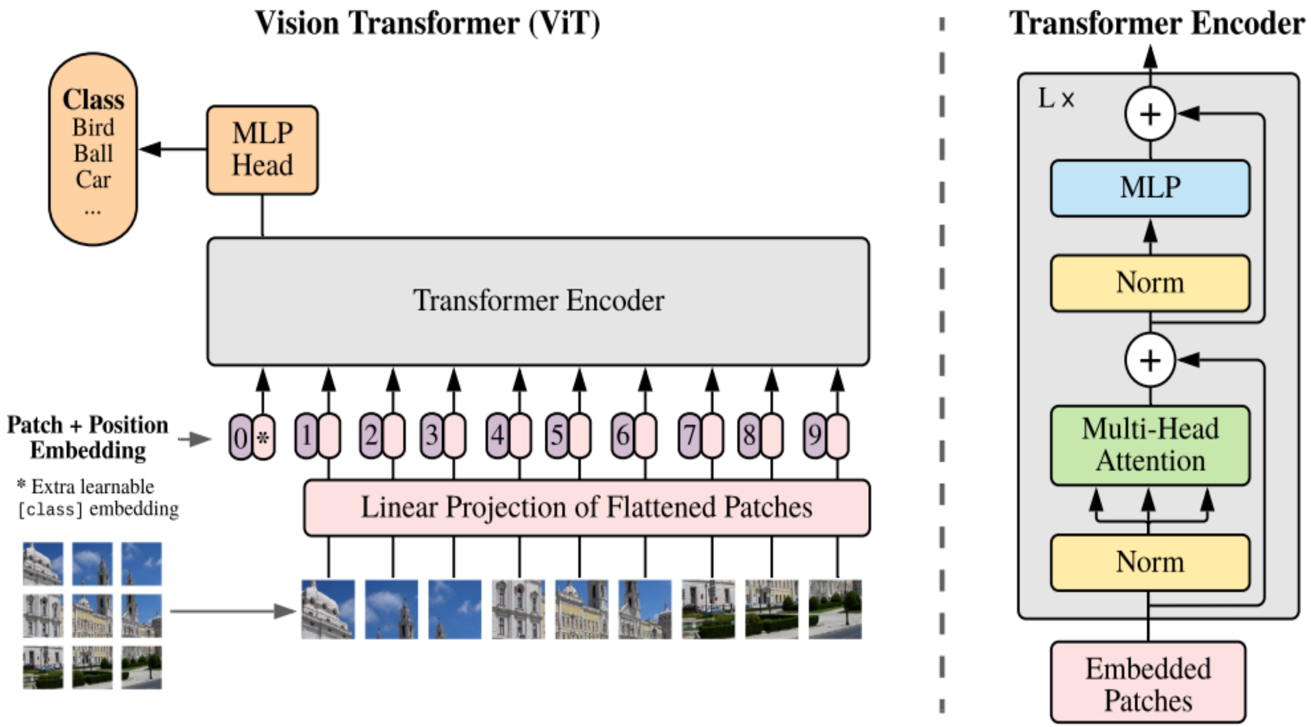
3) Repeat the process with multiple self-attention (head)
: concat and linear transform

4) → add & norm → MLP
→ add & norm

Transformers (ViT)

- Use only the encoder of the transformer
- Split the image into multiple patches (considered as one token)
- Flatten → Linear transformation → results in embedding vectors
- Slightly different order of application in the encoding blocks





DyTox Transformer Model

Setting

Symbol	Meaning
(x_i^t, y_i^t)	Input sample & its label from the t^{th} task
C^t	Label set of the t^{th} task
$C^{1:t}$	All labels from all seen tasks
$\boldsymbol{\theta}_t$	Task token of the t^{th} task
Clf_t	Independent classifier of the t^{th} task
SAB_l	l^{th} Self-Attention Block
TAB	Task-Attention Block

- At a given step $t \in \{1 \ldots T\}$,
- Training data : $\{(x_i^t, y_i^t)\}_i$
- All task label sets are exclusive : $C^0 \cap C^1 \ldots C^T = \emptyset$
- Only a few samples from previous tasks $\{1 \ldots t-1\}$ are available for training step t as **rehearsal** data.
- **Goal** of task t : classify test data coming from all seen classes $C^{1:t}$

DyTox Transformer Model

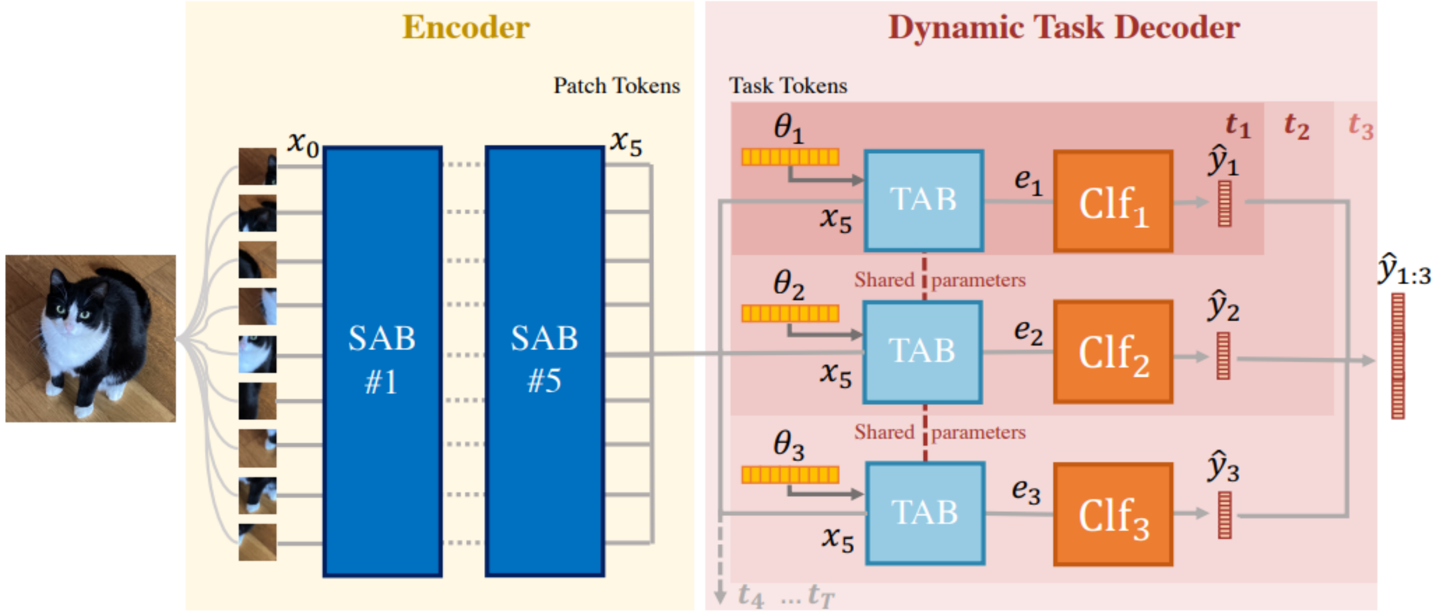


Figure 2: **DyTox transformer model**. An image is first split into multiple patches, embedded with a linear projection. The resulting patch tokens are processed by 5 successive Self-Attention Blocks (SAB) (Sec. 3.1). For each task ($t = 1 \dots T$), the processed patch tokens are then given to the Task-Attention Block (TAB) (Sec. 3.2): each forward through the TAB is modified by a different task-specialized token $\boldsymbol{\theta}_t$ for $t \in \{1 \dots T\}$ (Sec. 3.3). The T final embeddings are finally given separately to independent classifiers Clf_t each predicting their task's classes C^t. All $|C^{1:T}|$ logits are activated with a sigmoid. For example, at task $t = 3$, one forward is done through the SABs and three task-specific forwards through the unique TAB.

Encoder

- Patch tokenizer
- Self-Attention (SA) block

Decoder

- Task-Attention Block (TAB)
  - With task tokens
- classifier

Structure

- Patch tokenizer
  - fixed-size input RGB image

  → cropped into N patches

  → projected with a linear layer to a dimension D

  (these operations are done with a single 2D convolution)

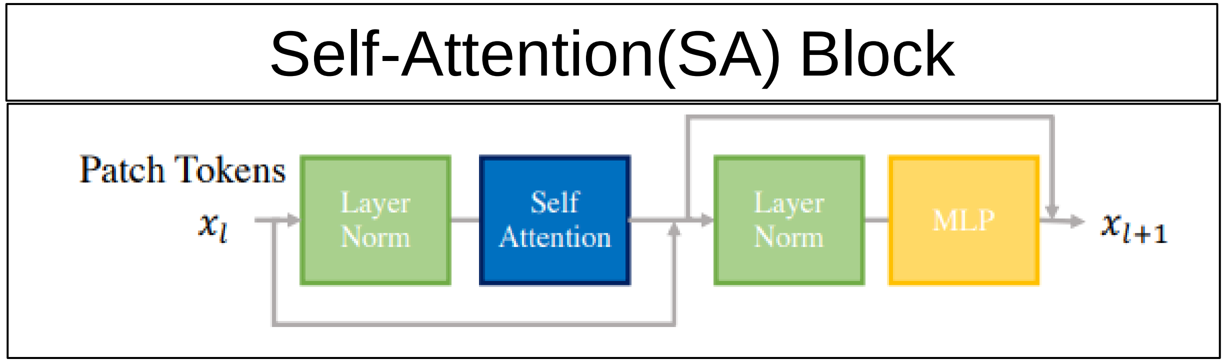
  → Result tensor : $x_0 \in \mathbb{R}^{N \times D}$

  + element-wise sum of positional embedding $p \in \mathbb{R}^{N \times D}$

  (In original ViT, classification token is concatenated to the result tensor

  → $[x_{cls}, x_0] \in \mathbb{R}^{(N+1) \times D}$)

Structure



- Self-Attention (SA) based encoder
  - $x_l' = x_l + \mathrm{SA}_l(\mathrm{Norm}_{l,1}(x_l))$
  - $x_{l+1} = x_l' + \mathrm{MLP}_l(\mathrm{Norm}_{l,2}(x_l'))$

Repeat from $l = 1$ to $l = L$ → result tensor : $x_L \in \mathbb{R}^{N \times D}$

Structure

- Task-Attention Block
  - In original ViT, "class token" is appended to the patch tokens after tokenizer and is given to a linear classifier with a softmax activation to predict final probabilities.
  - In this paper, **"task token"** (i^{th} task is denoted θ_i) is concatenated to the result of last SA block

$$z_i = [\theta_i, x_L] \in \mathbb{R}^{(N+1)\times D}$$

to exploit this task token, "Task-Attention" layer is newly defined.

Structure

- Task-Attention Block
  - Contrary to the classical Self-Attention, the Task-Attention defines its query (Q_i) only from the task-token θ_i without using the patch tokens x_L

Task-Attention

$Q_i = W_q\theta_i$
$K_i = W_k z_i$
$V_i = W_v z_i$
$A_i = Softmax\left(Q_i \cdot K_i^T / \sqrt{d/h}\right)$
$\boldsymbol{O_i} = W_o A_i V_i + b_o \in \mathbb{R}^{1\times D}$ (output)

d is the embedding dimension,
h is the number of attention heads.

Task-Attention Block

$c' = c + \mathrm{TA}(\mathrm{Norm}_1(z))$
$c'' = c' + \mathrm{MLP}(\mathrm{Norm}_2(c'))$

Summarization of all structure

$e_i = \mathrm{TAB} \circ ([\theta_i, \mathrm{SAB}_{l=L} \circ \ldots \mathrm{SAB}_{l=1}(x_0)]) \in \mathbb{R}^D$

$\tilde{y}_i = \mathrm{Clf}(e_i) = W_c \mathrm{Norm}_c(e_i) + b_c$

Structure

- Dynamic task token expansion
  - For given image x, through tokenizing and the multiple SABs, x_L is the output of Encoder.
  - In Decoder, each TAB forward passes is executed with a different task token θ_i → different task-specific forward
→ produce task-specific embedding : e_i

$$\begin{gathered} e_1 = \mathrm{TAB}([\boldsymbol{\theta_1}, x_L]) \\ e_2 = \mathrm{TAB}([\boldsymbol{\theta_2}, x_L]) \\ \dots \\ e_t = \mathrm{TAB}([\boldsymbol{\theta_t}, x_L]) \end{gathered}$$

  - Rather than concatenating all embeddings $\{e_1, e_2, \dots, e_t\}$, and make one classifier, **"task-specific classifiers"** are suggested.

Structure

- **task-specific classifiers :** $\hat{y}_i = \mathrm{Clf}_i(e_i) = \sigma(W_i \mathrm{Norm}_i e_i + b_i)$

- The predictions for the classes $\hat{y}_i \in C^i$, where σ is the sigmoid and loss is binary cross-entropy.

- In comparison with the softmax activation, the element-wise sigmoid activation reduces the overconfidence in recent classes → the model is better calibrated.

Structure

- The overall structure of the DyTox strategy

Algorithm 1 DyTox's forward pass at step t

Input: x_0 (initial patch tokens), y (ground-truth labels)
Output: $\hat{y}_{1:t}$ (predictions for all classes of $\mathcal{C}^{1:t}$)

$x_L \leftarrow \mathrm{SAB}_{l=L} \circ ... \mathrm{SAB}_{l=1}(x_0)$
for $i \leftarrow 1;\ \ i \leq t;\ \ i{+}{+}$ **do**
  $e_i \leftarrow \mathrm{TAB}([\boldsymbol{\theta}_i, x_L])$
  $\hat{y}_i \leftarrow \mathrm{Clf}_i(e_i)$
end for
$\hat{y}_{1:t} \leftarrow [\hat{y}_1, \ldots, \hat{y}_t]$

More efficient than a naïve parameter expansion network

- Memory overhead is almost null.
  The model's expansion is limited to a new task token per new task (in this case, only d=384 new parameters)
  ← very small compared to the total model size
  (≈11M parameters)

Structure

- Context
  - The Encoder is shared (both in memory and execution) for all outputs.
  - The Decoder parameters are also shared.
  - But its execution is task specific with each task token.
    (this is similar to a task-specific expert chosen from a mixture of experts)
  - Multi-task NLP-based transformers have natural language tokens as a task indicator (ex) summarization, translation,…), in the context of TAB, defined task tokens are used as indicators.

Structure

- Losses
  - The model is trained with three losses
  - $\mathcal{L}_{clf}$: classification loss (binary-cross entropy)
  - $\mathcal{L}_{kd}$: knowledge distillation (which is applied on the probability)
    - Helps to reduce forgetting
  - $\mathcal{L}_{div}$: divergence loss
    - Inspired from the auxiliary classifier of "DER"
    - Discriminate the current last task's classes C^t and all previous classes
    - Encourages a better diversity among task tokens.
    - Discarded at test-time

  - Total loss : $\mathcal{L} = (1 - \alpha)\mathcal{L}_{clf} + \alpha\mathcal{L}_{kd} + \lambda\mathcal{L}_{div}$
  - α correspond to the fraction of the number of old classes over the number of new classes $\frac{|C^{1:t-1}|}{|C^{1:t}|}$

Experiments : CIFAR100, ImageNet100 and ImageNet1000

- Implementation details
  - 5 Self-Attention Blocks (SABs)
  - 1 Task-Attention Blocks (TAB)
  - Each Attention Block has 12 heads and 384 dimension embedding

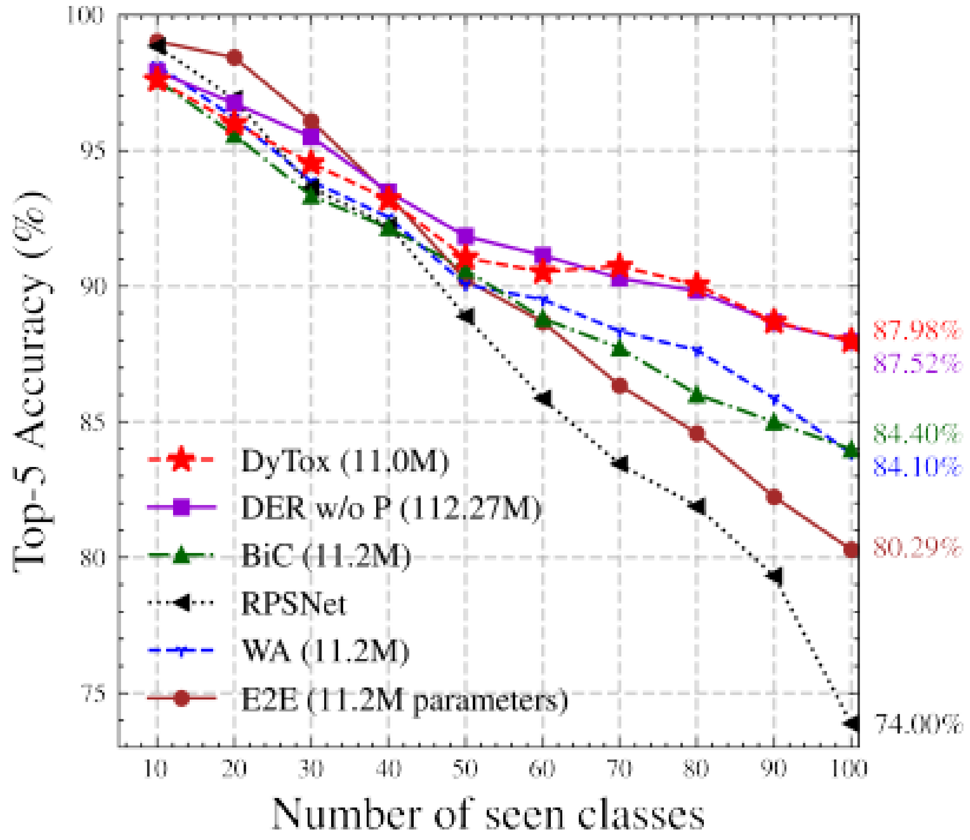
  - Hyperparameters are fixed for all experiments
    ← Use 10% of CIFAR 100's train set as validation and get hyperparameters.
    (apply to all experiments equally)

  - rehearsal memory : 2,000 images for CIFAR100 and ImageNet100, 20,000 images for ImageNet1000

Hyperparameter	CIFAR	ImageNet
# SAB	5	
# CAB	1	
# Attentions Heads	12	
Embed Dim	384	
Input Size	32	224
Patch Size	4	16

Table 1: **DyTox's architectures** for CIFAR and ImageNet. The only difference between the two architectures is the patch size, as the image sizes vary between datasets.

Experiments : Result of ImageNet100 and ImageNet1000



Methods	ImageNet100 10 steps #P	top-1 Avg	top-1 Last	top-5 Avg	top-5 Last	ImageNet1000 10 steps #P	top-1 Avg	top-1 Last	top-5 Avg	top-5 Last
ResNet18 joint	11.22	-	-	-	95.10	11.68	-	-	-	89.27
Transf. joint	11.00	-	79.12	-	93.48	11.35	-	73.58	-	90.60
E2E [5]	11.22	-	-	89.92	80.29	11.68	-	-	72.09	52.29
Simple-DER [48]	-	-	-	-	-	28.00	66.63	59.24	85.62	80.76
iCaRL [59]	11.22	-	-	83.60	63.80	11.68	38.40	22.70	63.70	44.00
BiC [32]	11.22	-	-	90.60	84.40	11.68	-	-	84.00	73.20
WA [81]	11.22	-	-	91.00	84.10	11.68	65.67	55.60	86.60	81.10
RPSNet [56]		-	-	87.90	74.00	-	-	-	-	-
DER w/o P [76]	112.27	**77.18**	66.70	**93.23**	87.52	116.89	68.84	60.16	88.17	82.86
DER† [76]	-	76.12	66.06	92.79	88.38	-	66.73	58.62	87.08	81.89
DyTox	11.01	77.15	69.10	92.04	87.98	11.36	71.29	63.34	88.59	84.49

The † marks the DER with setting specific pruning

Experiments : Result of CIFAR100

Methods	10 steps #P	10 steps Avg	10 steps Last	20 steps #P	20 steps Avg	20 steps Last	50 steps #P	50 steps Avg	50 steps Last
ResNet18 Joint	11.22	-	80.41	11.22	-	81.49	11.22	-	81.74
Transf. Joint	10.72	-	76.12	10.72	-	76.12	10.72	-	76.12
iCaRL [59]	11.22	65.27 ± 1.02	50.74	11.22	61.20 ± 0.83	43.75	11.22	56.08 ± 0.83	36.62
UCIR [32]	11.22	58.66 ± 0.71	43.39	11.22	58.17 ± 0.30	40.63	11.22	56.86 ± 0.83	37.09
BiC [75]	11.22	68.80 ± 1.20	53.54	11.22	66.48 ± 0.32	47.02	11.22	62.09 ± 0.85	41.04
WA [81]	11.22	69.46 ± 0.29	53.78	11.22	67.33 ± 0.15	47.31	11.22	64.32 ± 0.28	42.14
PODNet [19]	11.22	58.03 ± 1.27	41.05	11.22	53.97 ± 0.85	35.02	11.22	51.19 ± 1.02	32.99
RPSNet [56]	56.5	68.60	57.05	-	-	-	-	-	-
DER w/o P [76]	112.27	75.36 ± 0.36	65.22	224.55	74.09 ± 0.33	62.48	561.39	72.41 ± 0.36	59.08
DER† [76]	-	74.64 ± 0.28	64.35	-	73.98 ± 0.36	62.55	-	72.05 ± 0.55	59.76
DyTox	10.73	73.66 ± 0.02	60.67 ± 0.34	10.74	72.27 ± 0.18	56.32 ± 0.61	10.77	70.20 ± 0.16	52.34 ± 0.26
DyTox+	10.73	75.54 ± 0.10	62.06 ± 0.25	10.74	75.04 ± 0.11	60.03 ± 0.45	10.77	74.35 ± 0.05	57.09 ± 0.13

DyTox constantly surpasses previous state-of-the-art model, despite having a comparable performance at the first step and fewer parameters.

Experiments

- DyTox is able to scale correctly while handling seamlessly the parameter growth by sharing most of the weights across tasks.

- In contrast, DER had to propose a complex pruning method, and this pruning required different hyperparameter values for different settings.

- Despite this, the pruning in DER† is less efficient when classes diversity increase: DER† doubles in size between ImageNet100 and ImageNet1000 while handling the same amount of tasks (10).

Improved training procedure

- To bridge the gap between DyTox and DER w/o P on CIFAR100, a method called "MixUp" is used to create new samples from existing samples
  (DyTox+ : DyTox with MixUp)

- λ~Beta(α,α) (α=0.8)
  → new images : $x = \lambda x_1 + (1 - \lambda) x_2$
  as their labels : $y = \lambda y_1 + (1 - \lambda) y_2$
- Main Effect

1) It diversifies the training images and thus enlarges the training distribution in vicinity of each training sample
2) It improves the network calibration, reducing overconfidence in recent classes. (shared motivation with the sigmoid activation)

	1 step	50 steps	
Training	**Last** (↑)	**Last** (↑)	**Forgetting** (↓)
DyTox	76.12	52.34	33.15
DyTox+	77.51 +1.39	57.09 +4.75	31.50 -1.65

Table 4: **"Last" accuracy and forgetting** [8] on CIFAR100 for the joint (1 step, no continual) and 50 steps settings.

Model Introspection : on CIFAR100

- Memory overhead
  - Only add a vector of size d=384 per task → +0.004% per step (setting of CIFAR100 with 50 tasks, memory overhead was almost null (+0.2%)
- Model ablations
  - Knowledge distillation
  - Fine-tuning :
    - applied after each task on a balanced set of new data and rehearsal data
  - Token Expansion
  - Divergence Classifier
  - Independent Classifier

		Knowledge Distillation	Finetuning	Token Expansion	Divergence Classifier	Indendepent Classifiers	**Avg**	**Last**
DyTox	**Transformer**						60.69	38.87
		✓					61.62	39.35
		✓	✓				63.42	42.21
	Dynamic	✓	✓	✓			67.30	47.57
		✓	✓	✓	✓		68.28	49.45
		✓	✓	✓	✓	✓	70.20	52.34

CIFAR100 with 50 steps

Conclusion

- Self-attention layers are shared across all tasks, and we add task-specific tokens to achieve task-specialized embeddings through a new task-attention layer.

- It has very little memory overhead and does not require complex hyperparameter tuning.

- The experiments show that the DyTox framework scales to large datasets like ImageNet1k with state-of-the-art performances and number of parameters of the model increases reasonably contrary to previous dynamic strategies.