
CONTINUAL LEARNING WITH COLUMNAR SPIKING NEURAL NETWORKS

Denis Larionov
 Chuvash State University,
 Cheboksary, Russia
 Cifrum, Moscow, Russia
 denis.larionov@gmail.com

Nikolay Bazenkov
 Trapeznikov Institute of Control Sciences,
 Moscow Institute of Physics and Technology,
 Moscow, Russia

Mikhail Kiselev
 Chuvash State University,
 Cheboksary, Russia

June 23, 2025

ABSTRACT

This study investigates columnar-organized spiking neural networks (SNNs) for continual learning and catastrophic forgetting. Using CoLaNET (Columnar Layered Network), we show that micro-columns adapt most efficiently to new tasks when they lack shared structure with prior learning. We demonstrate how CoLaNET hyperparameters govern the trade-off between retaining old knowledge (stability) and acquiring new information (plasticity). Our optimal configuration learns ten sequential MNIST tasks effectively, maintaining 92% accuracy on each. It shows low forgetting, with only 4% performance degradation on the first task after training on nine subsequent tasks.

Keywords continual learning · catastrophic forgetting · spiking neural network · local learning

1 Introduction

Living organisms adapt continuously to changing environments. They accumulate, update, and utilize knowledge throughout their lives. Continual learning (CL) [1], often called incremental or lifelong learning [2], addresses this challenge in artificial intelligence.

Modern artificial neural networks (ANNs) assume stationary data distributions throughout training. In CL, this assumption fails—models must handle dynamically changing input distributions across sequential tasks. Training on new tasks often severely degrades performance on previous tasks, known as catastrophic forgetting (CF) [3, 4]. Backpropagation causes CF by requiring global parameter updates when learning new tasks.

Various approaches address CF in deep learning (Section 2). Biologically inspired methods use local learning, temporal dynamics, neuronal competition, and modulated plasticity. Spiking neural networks (SNNs) [5] provide a mathematical implementation of these principles.

However, [6] shows that local learning alone in SNNs is insufficient. The CF solution may require higher abstraction levels, considering functionally organized neuron groups rather than individual neurons. Cortical microcolumns exemplify such organization. CoLaNET architecture [7, 8, 9] presents an SNN based on columnar organization principles, combining columnar structure with local learning and other distinctive features. This combination should theoretically enable CL capabilities.

We investigate MNIST image classification using two experimental sets. First, we use the Permuted MNIST protocol from [10] to generate independent CL tasks. Second, we use a two-task protocol with Extended MNIST (EMNIST) [11], adding handwritten letters to classical MNIST. We compare against a 1-layer ANN baseline with 512 hidden neurons (similar to [10]).

In the Permuted MNIST ten-task experiment, CoLaNET shows perfect memory stability but sacrifices plasticity. By tuning microcolumn numbers, silent synapses, and adaptive threshold, CoLaNET achieves both low forgetting (4.35%

across 10 tasks) and decent average accuracy (92.31%). The source code and experiment results are open-source and available at: <https://gitflic.ru/project/dlarionov/cl>.

2 Related works

Several surveys cover continual learning research [12, 13]. Most recent studies focus on computer vision classification tasks [14]. Fewer works address text-based tasks [15], large language models [16], and reinforcement learning [17]. Common CL approaches fall into several classes.

Replay. Replay strategies approximate past task data distributions by storing training samples in memory buffers [18] or training generative models to reproduce past data. Interleaving old and new data during training helps retain knowledge of earlier tasks while learning new ones.

Regularization. These methods add penalty terms to preserve important parameters or behaviors from past tasks. Weight regularization restricts changes to critical parameters [19, 20], while function regularization maintains stable input-output mappings.

Optimization manipulation. This strategy adapts optimization itself for CL. Examples include seeking flatter minima through optimizer choice or learning rate scheduling, using adaptive per-synapse learning rates, projecting gradients to avoid interference, or employing fast/slow weights to separate short- and long-term updates.

Architectural modifications. These approaches dynamically adjust network structure for new tasks. Multi-head architectures assign separate output heads per task, while masking protects task-specific weights. Alternatives include gating mechanisms or dynamic expansion to isolate or partition knowledge.

Template-based classification. This approach uses class prototypes or generative models, creating templates (e.g., embedding-space prototypes) for each class and classifying new samples by proximity [14].

CF extends beyond mere forgetting [2]. First, CF represents the plasticity-stability dilemma: excessive learning plasticity interferes with memory stability, and vice versa [21]. Second, it relates to generalizability—distinguishing data distribution differences within and across tasks. Third, CF can be viewed as a computational resource optimization problem when training on new tasks. While reusing all past task data for new tasks would eliminate CF, it introduces high computational and storage costs along with potential privacy concerns. Training inefficiencies may also arise from excessive repetitions on single tasks (epochs). Thus, CL approaches typically restrict training to one epoch per task [22].

Nearly all successful CL strategies rely on gradient-based optimization, which causes CF. Local learning based on Hebbian plasticity offers an alternative. Enhanced with temporal dynamics, competition, gating mechanisms, and modulated plasticity, it provides a rich CL framework. Spiking neural networks (SNNs) [5] commonly implement these principles.

Relatively few studies explore SNNs for CL. Section 5.3 provides detailed comparison with [6].

3 Continual learning framework

3.1 Problem definition

Continual learning handles dynamically changing data distributions and task conditions. A model with parameters θ learns sequential tasks while preserving performance on older tasks after learning new ones.

Formally, training data for problem t is $\mathcal{D}_{t,b} = \{\mathbf{X}_{t,b}, \mathbf{Y}_{t,b}\}$ where $\mathbf{X}_{t,b}$ is input data, $\mathbf{Y}_{t,b}$ are labels, $t \in \mathcal{T} = \{1, \dots, k\}$ is the task index, and $b \in \mathcal{B}_t$ is the batch index. Here \mathcal{T} is the task space and \mathcal{B}_t is the batch space for task t . Tasks are defined by their training data \mathcal{D}_t . Data distribution within task $\mathcal{D}_t := p(\mathbf{X}_t, \mathbf{Y}_t)$ remains consistent between train and test sets. Labels \mathbf{Y}_t and task index t may be unavailable during training.

3.2 CL scenarios

The taxonomy in [12] categorizes CL scenarios using these concepts:

Task-based and task-free. Task-based problems present discrete tasks sequentially with clear boundaries. Task-free scenarios have discrete tasks but soft transitions.

Task-, domain-, and class-incremental. Task-incremental learning (TIL) learns distinct tasks with always-available task identifiers. Domain-incremental learning (DIL) maintains consistent task structure (label space) but changes input data

context over time, making context determination impossible. Class-incremental learning (CIL) has disjoint label spaces per task, with task identifiers available only during training.

Streaming. Each example appears only once.

Online. Only one example appears at a time.

The experiments conducted in the study belong to the class of online streaming task-based domain incremental learning.

3.3 Evaluation metrics

3.3.1 Overall performance

We evaluate overall performance using average accuracy (AA) and average incremental accuracy (AIA). Let $a_{k,j} \in [0, 1]$ be classification accuracy on task j after learning task k . Average accuracy is:

$$AA_k = \frac{1}{k} \sum_{j=1}^k a_{k,j}. \quad (1)$$

Average incremental accuracy is:

$$AIA_k = \frac{1}{k} \sum_{i=1}^k AA_i. \quad (2)$$

AA reflects current overall performance, while AIA aggregates performance across previous tasks.

3.3.2 Memory stability

We evaluate memory stability using forgetting measure (FM) and backward transfer (BWT). FM of task j after training on task k is the difference between maximum past performance and current performance:

$$f_{j,k} = \max_{i \in \{1, \dots, k-1\}} (a_{i,j} - a_{k,j}), \quad \forall j < k. \quad (3)$$

FM of task k is average forgetting across previous tasks:

$$FM_k = \frac{1}{k-1} \sum_{j=1}^{k-1} f_{j,k}. \quad (4)$$

BWT is the average influence of learning task k on previous tasks:

$$BWT_k = \frac{1}{k-1} \sum_{j=1}^{k-1} (a_{k,j} - a_{j,j}), \quad (5)$$

where negative BWT values indicate forgetting.

3.3.3 Generalization ability

We evaluate generalization using forward transfer (FWT), the average influence of previous tasks on task k :

$$FWT_k = \frac{1}{k-1} \sum_{j=2}^k (a_{j,j} - \tilde{a}_j), \quad (6)$$

where \tilde{a}_j is baseline model accuracy on task j .

4 Columnar spiking network

CoLaNET (Columnar Layered Network) architecture [7, 8, 9] embodies columnar organization in SNNs with local learning, neuronal competition, modulated plasticity, and gating mechanisms. Designed for supervised classification, CoLaNET’s key feature integrates anti-Hebbian plasticity (degrading weights) with modulated plasticity (strengthening weights), counteracting degradation and enabling effective learning.

4.1 Network architecture

CoLaNET consists of multiple identical modules (columns), with column count matching class count. Each column has five layers: trainable input, three intermediate processing, and output layers. The lower three layers organize into microcolumns - small interconnected neuron groups recognizing subclasses within single classes.

CoLaNET training relies on microcolumn competition for activation when presented with input patterns. Initially requiring external guidance for correct neuron activation, synaptic adjustments gradually strengthen relevant connections while weakening incorrect ones. A reward mechanism reinforces synapses in correct classifications, while anti-Hebbian learning suppresses erroneous activations, ensuring stable, efficient learning [7].

In [9], genetic algorithm hyperparameter optimization yields a CoLaNET configuration with 15 separate networks for MNIST classification. Each network contains 10 columns (matching class count) with 15 microcolumns each. Total neuron count is 7,050, with 2,250 having plastic synapses. This configuration achieves 95% accuracy in single-epoch training, while individual ensemble networks reach only 75%.

CoLaNET converts pixel intensities to spike sequences using rate coding, where spike generation probability at each time step equals pixel intensity ratio (0 intensity = zero probability, 255 = maximum probability). Images appear for 10 time steps, followed by 10 silence steps.

All our experiments use the MNIST-optimized CoLaNET configuration from [8].

4.2 CL features

CoLaNET’s adaptive firing threshold balances learning plasticity and memory stability [7]. The adaptive threshold u_{tr} combines a constant component with a term proportional to positive weight sum:

$$u_{tr} = u_{const} + \alpha \sum_i w_i^+, \quad (7)$$

where α is the proportionality coefficient, u_{const} is the fixed threshold part, and $\sum_i w_i^+$ is the positive weight sum ($w_i > 0$).

Parameter α controls neuron specialization degree for specific subclasses. Strong synapses increase firing threshold, enabling activation only for specific input patterns.

Synaptic resource renormalization [8] also influences neuron specialization. Strengthened synapses proportionally weaken others, and vice versa. CoLaNET disables renormalization by default but enables it by specifying virtual synapse count (ns) that absorbs parameter changes. When $ns = 0$, renormalization effect maximizes; increasing ns diminishes it to zero.

Microcolumn count per column also controls network capacity, crucial for scaling task numbers.

4.3 ArNI-X framework

We implemented all SNN training and evaluation experiments using ArNI-X framework [23]. Since CoLaNET was originally designed in ArNI-X, the framework includes all necessary CoLaNET mechanics.

5 Experiments

5.1 Permuted MNIST

5.1.1 Tasks protocol

Permuted MNIST is a popular CL framework with unlimited tasks [20, 24, 18]. Each task uses random pixel permutation of MNIST dataset. Permutation remains consistent across all task images. We evaluate models on all previous tasks [10]. Figure 1 shows example permuted images.

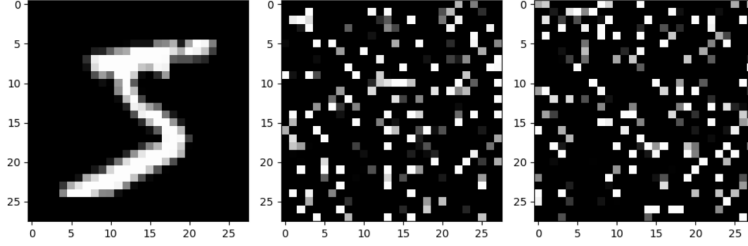


Figure 1: Two random pixel permutations of digit 5.

Random pixel permutations destroy spatial patterns, preventing CNN use and human classification. However, fully connected networks (like CoLaNET) find transformed images equally informative.

Sequential learning across ten MNIST permutations creates a degradation profile—a 10×10 matrix where element $a_{k,j} \in [0, 1]$ represents classification accuracy on task j 's test set after learning task k ($j \leq k$). From this profile we compute Average Accuracy (AA), Average Incremental Accuracy (AIA), Forgetting Measure (FM) and Backward Transfer (BWT) (Section 3.3).

5.1.2 Baseline model

We trained a fully connected ANN with 512 neurons in the hidden layer and 10 neurons in the output layer for classification as a baseline model [25]. The network uses SGD with adaptive learning rate (AdaDelta) and achieves 98% accuracy on the MNIST test set after 5 epochs. For CL, we set one epoch per task. This slightly reduces individual task performance (98% to 96%) but reduces inter-task interference and maintains online learning.

Table 1 shows the baseline degradation profile. Columns represent tasks, rows represent training iterations on each task. Each cell contains test set classification accuracy for the respective task. Table 3 presents final computed metrics.

Table 1: Baseline model. Columns are task identifiers, rows are training iterations.

96.81	15.33	5.77	8.62	8.06	11.72	5.91	9.00	9.46	6.08
94.82	96.52	4.77	6.68	14.72	12.16	6.35	9.91	9.47	6.92
89.14	95.45	96.83	9.02	19.31	13.87	5.54	9.36	10.27	4.13
81.73	93.93	95.58	96.50	16.28	14.07	7.37	9.98	9.50	6.96
79.35	91.68	94.18	95.02	96.25	15.94	8.90	10.78	11.07	8.68
71.54	86.79	92.13	91.67	95.10	96.18	7.99	9.12	9.67	8.90
71.79	79.97	85.93	90.04	93.06	94.92	96.71	8.90	13.11	9.18
67.39	73.21	80.01	84.26	89.49	92.07	95.16	96.02	13.64	9.73
58.18	65.47	71.49	79.13	83.41	87.16	93.53	95.37	95.85	11.30
49.43	55.05	60.97	70.72	76.77	80.90	87.52	93.46	93.93	95.45

5.1.3 CoLaNET

We implemented Permuted MNIST experiments with CoLaNET in three stages. Stage one uses the CoLaNET configuration from [8] (Section 4), optimized for single-task performance without considering CL specifics like stability-plasticity balance.

We run ArNI-X sequentially across 10 Permuted MNIST tasks, then evaluate accuracy on all previous tasks (55 total runs per experiment). Each training iteration generates a network state file for evaluating all previous tasks. From

iteration two onward, the network loads state from the previous task. CoLaNET experiments on dual Quadro RTX 6000 systems take 1h 20m for 15 microcolumns and 3h for 45 microcolumns.

Table 2 shows results using optimal CoLaNET configuration from [8]: first-task accuracy meets expectations (94%). Then significant performance drops occur: 10% decrease by Task 2 (83%), and 50% by Task 3. Notably, the network shows zero forgetting: task accuracies remain constant throughout training.

Table 2: CoLaNET degradation profile for 15 microcolumns and $\alpha = 0.023817$.

94.36									
94.34	83.67								
94.34	83.61	34.17							
94.34	83.61	34.68	36.75						
94.33	83.60	34.68	36.76	25.59					
94.45	83.60	34.68	36.76	25.59	23.97				
94.45	83.60	34.68	36.76	25.59	23.97	12.01			
94.43	83.60	34.68	36.76	25.59	23.97	12.03	15.25		
94.43	83.60	34.68	36.76	25.59	23.97	12.03	15.27	11.87	
94.43	83.60	34.68	36.76	25.59	23.97	12.03	15.27	11.87	13.55

Table 3: Overall performance on Permuted MNIST

Baseline model (1-layer ANN)	AA	AIA	FM	BWT
avg	88.80	29.39	9.17	-9.17
max	96.81	47.11	22.10	-22.10
std	6.19	12.05	6.53	6.53
CoLaNET (15 microcolumns, $\alpha = 0.023817$)				
avg	57.92	19.86	0.00	0.05
max	94.36	26.35	0.04	-0.04
std	19.89	5.20	0.04	0.05
CoLaNET (15 microcolumns, $\alpha = 0.005$)				
avg	89.50	29.12	2.88	-2.88
max	91.03	49.03	3.47	-3.47
std	0.73	12.73	1.13	1.13
CoLaNET (45 microcolumns, $\alpha = 0.1$)				
avg	92.31	30.02	1.19	-1.18
max	93.30	50.67	1.49	-1.49
std	0.39	13.12	0.41	0.40

Results show balance shifted toward memory stability at plasticity’s expense. The absence of forgetting suggests that microcolumns specialized for one task stop adapting to subsequent tasks entirely. After training on the first task, the network still retains a portion of microcolumns with low-magnitude weights capable of learning. However, this pool shrinks with each new task. Eventually the network cannot maintain high accuracy on novel tasks, with performance dropping to near random (10%).

One mechanism for controlling the degree of neuron specialization is the adaptive threshold. It increases the activation threshold proportionally to the sum of positive weights, scaled by coefficient α (Equation 7). In CoLaNET’s optimal configuration, $\alpha = 0.023817$ [8].

Stage two uses varying α values from 0 to 0.023817. When $\alpha = 0$, the threshold loses adaptive properties, degrading classification accuracy from 94% to 90%. However, this enables full-capacity learning on each new task. Results again show skewed balance, now favoring learning plasticity over memory stability. Figure 2 presents AA and FM metrics for different α values. For $\alpha > 0.015$, the network almost stops forgetting. Optimal value is $\alpha = 0.005$, with degradation profile in Table 4 and metrics in Table 3. Learning plasticity degrades for $\alpha > 0.005$ as network capacity proves insufficient for new tasks. More microcolumns in CoLaNET configuration can improve this.

In the third stage, we repeated the Permuted MNIST experiment for selected α values with triple the number of microcolumns (from 15 to 45 microcolumns per column). We found the optimal α value for the 45-microcolumn configuration to be 0.01, which maintains learning plasticity across all 10 tasks (Table 3). Table 5 shows the degradation profile of CoLaNET with $\alpha = 0.01$ and 45 microcolumns.

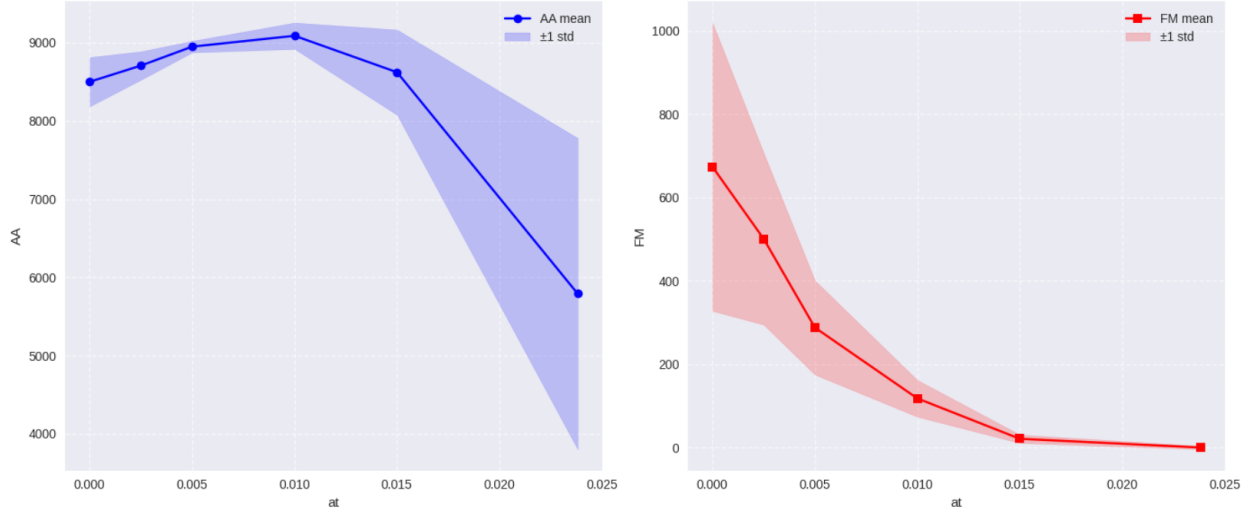


Figure 2: Average Accuracy and Forgetting Measure with different α for CoLaNET with 15 microcolumns per column.

Table 4: CoLaNET degradation profile for 15 microcolumns and $\alpha = 0.005$.

91.03									
89.35	91.68								
85.62	90.60	92.29							
85.13	90.54	91.90	92.49						
82.65	88.15	90.85	92.01	93.06					
80.41	87.83	90.41	91.14	92.77	92.05				
80.06	87.21	89.72	91.02	92.10	91.78	91.94			
78.85	86.55	89.29	90.58	91.08	91.48	91.89	91.99		
78.68	84.51	88.58	89.87	90.76	90.91	91.54	91.74	91.94	
78.42	84.00	88.45	89.38	90.14	90.69	91.42	91.59	91.63	91.56

The CoLaNET configuration in Table 5 learns effectively across ten tasks and resists forgetting well, degrading only $93.30 - 88.95 = 4.35\%$ on the first task after learning nine subsequent tasks (compared to 49% degradation for the baseline model).

Permuted MNIST simplifies stability-plasticity trade-off analysis in CoLaNET by eliminating inter-task interference through pattern independence, but this artificial separation must be explicitly addressed in real-world scenarios where tasks share underlying common features.

5.2 Two tasks based on E/MNIST

5.2.1 EMNIST Dataset

The EMNIST dataset [11] extends MNIST by including both handwritten digits (0-9) and English alphabet letters (A-Z, a-z). Following [6], we use the Balanced subset with ten selected letter classes (A, B, D, E, G, H, N, Q, R, S), resulting in 24k training and 4k test images. For two-task experiments conducted bidirectionally, we balance the tasks by using only the first 28k images from original MNIST (24k training, 4k test).

5.2.2 Baseline model

We trained the identical fully connected network (512h, 10o) from Section 5.1.2 as our baseline, achieving 95%/93% accuracy on MNIST/EMNIST respectively [26] on 1 epoch. The results reveal significant (75%/69%) forgetting in both task directions, contrasting sharply with the minimal forgetting observed in Permuted MNIST experiments (Table 1). This substantial difference stems from the intrinsic similarity between MNIST and EMNIST images (centered character/letter patterns), which creates shared feature representations across tasks.

Table 5: CoLaNET degradation profile for 45 microcolumns and $\alpha = 0.01$.

93.30									
91.84	93.23								
91.03	92.51	93.62							
90.48	92.01	93.45	93.87						
90.19	91.86	93.02	93.48	92.97					
89.75	91.69	92.88	93.22	92.86	93.01				
89.57	91.36	92.82	92.95	92.79	92.99	93.03			
89.09	91.20	92.69	92.60	92.62	92.99	93.27	91.73		
89.00	91.19	92.46	92.57	92.47	92.92	92.96	91.69	92.35	
88.95	91.29	92.37	92.37	92.31	92.65	92.75	91.43	92.30	91.22

5.2.3 CoLaNET

We conducted the E/MNIST two-task experiments with CoLaNET in two stages. Stage one tests the CoLaNET configuration from [8], sweeping α values in Equation 7 from 0 to 0.03.

For the experimental pipeline, we ran ArNI-X (Section 4.3) sequentially on both tasks (forward/reverse order) plus an additional evaluation run on the first task (6 runs total). Each experiment completed within 10 minutes on a dual Quadro RTX 6000 24GB setup.

Results show that decreasing α increased forgetting (reaching 60% at $\alpha = 0$, similar to a basic fully connected network). Increasing α slightly reduced forgetting but severely hurt EMNIST performance. This happens because the tasks share features—microcolumns trained on one task also respond to the other, especially when the activation threshold is low.

Table 6: CoLaNET results on E/MNIST. Accuracy values are reported as whole numbers by omitting the decimal point (e.g., 88.67% is written as 8867).

Parameters α and ns	MNIST →EMNIST	MNIST	FM1	EMNIST →MNIST	EMNIST	FM2
0	8867→8250	2509	6358	7972→8865	1927	6045
0.00125	9265→8245	3232	6033	8057→9137	3092	4965
0.00125, 0	9087→8772	4070	5017	8720→9272	3577	5143
0.00125, 1k	9187→8750	4340	4847	8725→9270	4480	4245
0.00125, 10k	9207→8775	3822	5385	8640→9275	4467	4173
0.00125, 100k	9115→8717	4039	5076	8662→9345	4419	4243
0.0015	9240→8292	3990	5250	8115→9010	3557	4558
0.0015, 0	9115→8745	4122	4993	8707→9262	3912	4795
0.0015, 1k	9197→8790	4182	5015	8822→9317	4392	4430
0.0015, 10k	9115→8795	4167	4948	8670→9362	4685	3985
0.0015, 100k	9270→8835	4290	4980	8810→9352	4877	3933
0.002	9260→8110	4902	4358	8010→8947	3805	4205
0.002, 0	9185→8112	4419	4766	8695→7485	5240	3455
0.002, 1k	9230→8665	4447	4783	8897→8990	5299	3598
0.002, 10k	9192→8602	4202	4990	8850→9120	5230	3620
0.002, 100k	9235→8767	4480	4755	8850→9190	5342	3508
0.00238	9242→7835	5687	3555	7849→8665	3837	4012
0.00238, 0	9110→6552	5275	3835	8325→2970	5595	2730
0.00238, 1k	9220→8160	5022	4198	8757→8095	5447	3310
0.00238, 10k	9247→7964	5302	3945	8807→8670	5802	3005
0.00238, 100k	9290→8425	4950	4340	8840→8772	5635	3205
0.003	9212→6395	6655	2557	7454→8212	4024	3430
0.003, 0	9172→2655	6100	3072	6917→444	4407	2510
0.003, 1k	9335→5977	6372	2963	8220→3665	5472	2748
0.003, 10k	9160→4419	6687	2473	8592→6762	5647	2945

In the second stage of the two-task experiment, we tested the CoLaNET configuration with different numbers of virtual synapses (parameter ns , ranging from 0 to 100k) for various α values. Table 6 presents the experimental results. The FM (forgetting measure) column shows the difference between the accuracy on the first task before and after training on the second task. Specifically, the experiments demonstrate that classification accuracy on EMNIST increases

significantly either with a slight decrease in α or with the introduction of minor renormalization (10–100k virtual synapses).

The best CoLaNET configuration maintains relatively high post-training accuracy (above 87% on all tasks) while achieving relatively low forgetting (43% for MNIST→EMNIST and 32% for EMNIST→MNIST). This configuration uses $\alpha = 0.023817$ and $ns = 100k$. Although sacrificing some accuracy can reduce forgetting to 25%, even this level of forgetting remains unsatisfactory in the two-task experiment.

5.2.4 Heatmaps

Since CoLaNET is a fully connected network, visualizing its receptive fields becomes a convenient analysis tool. Figure 3 displays the receptive fields (as heatmaps) of plastic neurons in CoLaNET after training on two tasks sequentially (first MNIST, then EMNIST). We observe that the columns combine patterns from both MNIST digits and EMNIST letters. For example, roughly half of the microcolumns in the first row contain a prototype of the digit 0, while the other half show a prototype of the letter A. In the second row, the prototype of digit 1 appears only in the first microcolumn, while most others exhibit the prototype of letter B. In some columns, a small number of microcolumns experience interference between the two tasks, leading to a loss of distinct patterns—however, the vast majority of microcolumns remain specialized for a single pattern.

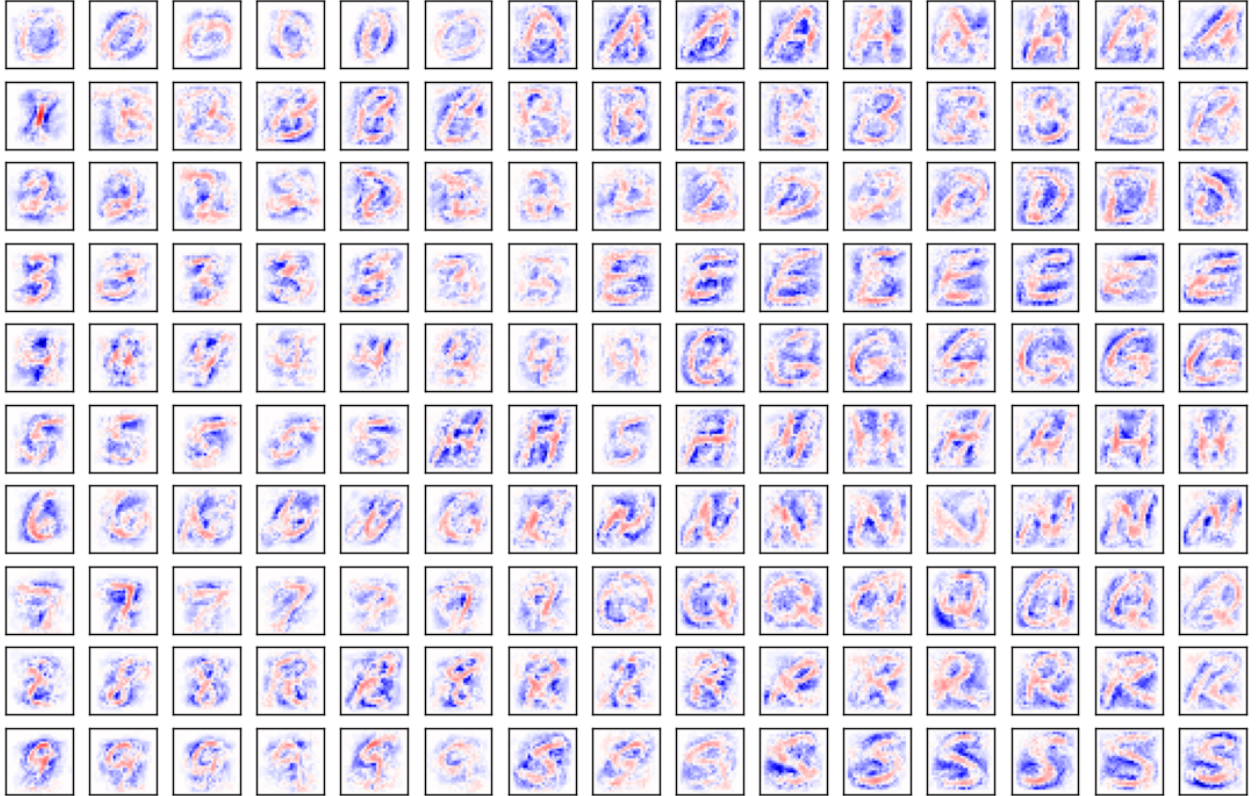


Figure 3: Receptive fields of plastic neurons after training on MNIST and subsequent training on EMNIST. Blue indicates negative weights; red indicates positive weights

5.3 Comparison of the results

Comparing CL methods is challenging due to varying datasets, problem formulations, evaluation metrics, task counts, training epochs, label assumptions, and architectures. For example, Synaptic Intelligence (SI) [20] uses 20 epochs on Permuted MNIST, while Elastic Weight Consolidation (EWC) [19] employs early stopping.

[22] compares replay methods on Permuted MNIST using a two-layer network similar to the one discussed in Section 5.1.2 in an online setting (one epoch). Table 7 shows CoLaNET outperforms these methods (accounting for architectural differences).

Table 7: Comparison of the effectiveness of different replay approaches with a buffer size of 50 elements on ten Permuted MNIST tasks, using one training epoch per task. The bottom section presents similar results for CoLaNET in configurations with 15 and 45 microcolumns. The results are averaged over ten independent experiments.

	AA	FM
Joint training	89.05 ± 0.27	
GEM [24]	74.57 ± 0.10	7.40 ± 0.11
AGEM [27]	69.50 ± 0.76	13.10 ± 0.63
MER [28]	75.75 ± 0.65	8.74 ± 0.73
MIR [29]	78.31 ± 0.63	7.15 ± 0.67
CTN [30]	79.70 ± 0.44	5.08 ± 0.44
NCC [22]	83.47 ± 0.43	3.44 ± 0.26
CoLaNET (15 microcolumns)	89.67 ± 0.16	2.75 ± 0.23
CoLaNET (45 microcolumns)	92.39 ± 0.13	0.94 ± 0.17

For E/MNIST, CoLaNET uses the same task set as [6] (28000 examples, 10 classes), but differs in architecture, plasticity mechanics, initialization (random values), and input encoding (DOG). Table 8 compares their results.

Table 8: Comparison of the effectiveness of different SNN-based approaches to transfer learning (MNIST→EMNIST) and their matching with analogous results for CoLaNET.

	MNIST→EMNIST	MNIST	FM
SNN [31]	90.8→78.4	48.1	42.7
Lateral inhibition	94.8→88.9	78.6	16.2
Pseudo-rehearsal	93.6→79.0	43.5	50.1
Self-reminder (0.25%)	93.6→74.0	74.5	19.1
Self-reminder (10%)	93.6→77.8	91.1	2.5
Noise regularization	93.9→87.8	67.2	26.7
Dropout	94.2→87.6	62.9	31.3
Frozen large weights	93.6→69.2	78.2	15.4
Langevin dynamics	93.6→78.3	82.2	11.4
Joint training	93.6→79.7	92.0	1.6
CoLaNET	92.9→84.3	49.5	43.4

The original SNN experiments in [6] show relatively high forgetting (42%), comparable to CoLaNET (43%). Few-shot self-reminder (memorizing 10% of initial examples) proved their best mitigation strategy, while regularization, dropout, weight freezing, and pseudo-rehearsal proved ineffective.

[6] notes that interference between tasks, which leads to relatively high forgetting, occurs primarily at the deepest (third) level of the hierarchy. An experiment demonstrates this by freezing the weights of the first two layers during training on the second EMNIST task (performance on MNIST dropped by less than 1% compared to the non-frozen version). Assuming that feature representations become less general at deeper hierarchical levels (discussed in Section 6), the CoLaNET architecture can effectively address this problem.

6 Conclusion

When tasks lack commonality, the optimal CoLaNET configuration remains fully resistant to forgetting in the DIL setting across many tasks. However, this configuration struggles to learn new tasks. Allowing slight forgetting (under 5% across 10 tasks) lets CoLaNET learn new tasks effectively (with only a 1% performance drop compared to the optimal setup).

We can control the balance between plasticity and memory stability in CoLaNET through microcolumn specialization (determined by parameter α in Equation 7) and the number of virtual synapses (ns), which divert synaptic resource renormalization. Adding capacity by increasing microcolumns per column is possible but leads to network growth and computational overhead.

Permuted MNIST results do not fully generalize to E/MNIST due to shared features — MNIST digits resemble EMNIST letters, increasing interference as microcolumns respond to similar patterns.

Effective continual learning for tasks with shared features may require hierarchical architecture, applying CoLaNET only to deep layers. Shared features dominate shallow layers, while deeper layers exhibit task-specific representations. For example, the primary visual cortex (V1) detects low-level features (edges, lines) with high commonality, while deeper areas (V2, V4) combine them into complex, task-specific representations. Similarly, CNNs first learn general textures, then geometric shapes, and finally semantic features with minimal overlap. Hierarchy also improves capacity efficiency: shallow layers store shared knowledge, enabling FWT and avoiding redundant parameter replication.

In summary, columnar-organized SNNs provide a promising foundation for CL solutions. By combining local learning, competition, modulation, and gating mechanisms, they achieve parameter separation between tasks, balancing high plasticity and memory stability — particularly for non-overlapping feature representations, which characterize deeper layers.

References

- [1] Mark B. Ring. Child: A first step towards continual learning. *Machine Learning*, 28(1):77–104, 1997.
- [2] Dhireesha Kudithipudi and et. al. Biological underpinnings of lifelong learning machines. *Nature Machine Intelligence*, 4, 2022.
- [3] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*, 24:109–165, 1989.
- [4] Roger Ratcliff. Connectionist models of recognition memory: Constraints imposed by learning and forgetting functions. *Psychological Review*, 97(2):285–308, 1990.
- [5] Mikhail Kiselev. *Spike Neural Networks: Information Representation, Learning, Memory*. Palmarium Academic Publishing, 2020.
- [6] D.I. Antonov, K.V. Sviatov, and S. Sukhov. Continuous learning of spiking networks trained with local rules. *Neural Networks*, 155:512–522, 2022.
- [7] Mikhail Kiselev. Colanet – a spiking neural network with columnar layered architecture for classification. *arXiv preprint arXiv:2409.01230*, 2024.
- [8] Mikhail Kiselev. Classifying images with colanet spiking neural network - the mnist example. *arXiv preprint arXiv:2409.07833*, 2024.
- [9] Mikhail Kiselev. A digital machine learning algorithm simulating spiking neural network CoLaNET. *arXiv preprint arXiv:2503.17111*, 2025.
- [10] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.
- [11] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. Emnist: an extension of mnist to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017.
- [12] Gido M. van de Ven, Nicholas Soures, and Dhireesha Kudithipudi. Continual learning and catastrophic forgetting. *arXiv preprint arXiv:12403.05175*, 2024.
- [13] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, 2024.
- [14] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(7):3366–3385, 2022.
- [15] Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costajussa. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541, 2020.
- [16] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey. *ArXiv*, abs/2402.01364, 2024.
- [17] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *Journal of Artificial Intelligence Research*, 75:1401–1476, 2022.
- [18] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet Kumar Dokania, Philip H. S. Torr, and Marc’Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv: Learning*, 2019.

- [19] James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114, 2016.
- [20] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. *Proceedings of machine learning research*, 70, 2017.
- [21] Stephen Grossberg. Processing of expected and unexpected events during conditioning and attention: a psychophysiological theory. *Psychological Review*, 89(5):529–572, 1982.
- [22] Haiyan Yin and Peng Yang. Mitigating forgetting in online continual learning with neuron calibration. In *Neural Information Processing Systems*, 2021.
- [23] Mikhail Kiselev. Arni-x: A simple but powerful and flexible simulator for spiking neural networks, 2020.
- [24] David Lopez-Paz and Marc’Aurelio Ranzato. Gradient episodic memory for continual learning. In *Neural Information Processing Systems*, 2017.
- [25] Denis Larionov. Continual learning on permuted MNIST, 2024.
- [26] Denis Larionov. Continual learning on EMNST, 2025.
- [27] Arslan Chaudhry, Marc’Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *ArXiv*, 2018.
- [28] Matthew Riemer, Ignacio Cases, Robert Ajemian, Miao Liu, Irina Rish, Yuhai Tu, and Gerald Tesauro. Learning to learn without forgetting by maximizing transfer and minimizing interference. *ArXiv*, 2018.
- [29] Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *ArXiv*, 2019.
- [30] Quang Hong Pham, Chenghao Liu, Doyen Sahoo, and Steven C. H. Hoi. Contextual transformation networks for online continual learning. In *International Conference on Learning Representations*, 2021.
- [31] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, and Timothée Masquelier. Spyketorch: Efficient simulation of convolutional spiking neural networks with at most one spike per neuron. *Frontiers in Neuroscience*, 13, 2019.