Domain-Agnostic Neural Architecture for Class Incremental Continual Learning in Document Processing Platform

Mateusz Wójcik[1,2], **Witold Kościukiewicz**[1,2], **Mateusz Baran**[1,2],
Tomasz Kajdanowicz[1], **Adam Gonczarek**[2]
[1]Wroclaw University of Science and Technology [2]Alphamoon Ltd., Wrocław
{mateusz.wojcik,tomasz.kajdanowicz}@pwr.edu.pl
adam.gonczarek@alphamoon.ai

Abstract

Production deployments in complex systems require ML architectures to be highly efficient and usable against multiple tasks. Particularly demanding are classification problems in which data arrives in a streaming fashion and each class is presented separately. Recent methods with stochastic gradient learning have been shown to struggle in such setups or have limitations like memory buffers, and being restricted to specific domains that disable its usage in real-world scenarios. For this reason, we present a fully differentiable architecture based on the Mixture of Experts model, that enables the training of high-performance classifiers when examples from each class are presented separately. We conducted exhaustive experiments that proved its applicability in various domains and ability to learn online in production environments. The proposed technique achieves SOTA results without a memory buffer and clearly outperforms the reference methods.

1 Introduction

Solutions based on deep neural networks have already found their applications in almost every domain that can be automated. An essential part of them is NLP, the development of which has gained particular momentum with the beginning of the era of transformers (Vaswani et al., 2017). Complex and powerful models made it possible to solve problems such as text classification with a previously unattainable accuracy. However, exploiting the capabilities of such architectures in real-world systems requires online learning after deployment. This is especially difficult in dynamically changing environments that require the models to be frequently retrained due to domain or class setup shifts. An example of such environment is Alphamoon Workspace[1] where the presented architecture will be deployed as a model for document classification since we noticed the emerging need for online learning. We observed that the users' data in document classification process is changing frequently and such shifts often decrease the model accuracy. As a result, we have to retrain the models manually ensuing a time-consuming process. Our goal was to design an effective approach to incremental learning that will be used in a continual learning module of our system (Figure 1).

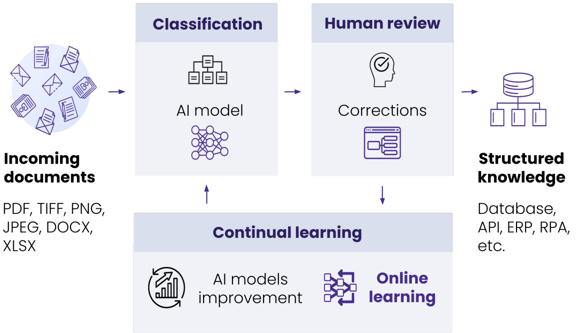

Figure 1: Continual learning in document processing platform. Classification models need to learn incrementally and handle domain shifts after deployment.

Recently, neural architectures have become effective and widely used in classification problems (Devlin et al., 2018; Rawat and Wang, 2017). The parameter optimization process based on gradient descent works well when the data set is sufficiently large and fully available during the training process. Otherwise, the catastrophic forgetting (French, 1999) may occur, which makes neural networks unable to be trained incrementally. Continual learning aims to develop methods that enable accumulating new knowledge without forgetting previously learnt one.

In this paper, we present a domain-agnostic architecture for online class incremental continual learning called DE&E (Deep Encoders and Ensembles). Inspired by the E&E method (Shanahan et al., 2021), we proposed a method that increases its accuracy, provides full differentiability, and, most

[1] https://alphamoon.ai/

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 5: Industry Track, pages 527–537
July 10-12, 2023 ©2023 Association for Computational Linguistics

importantly, can effectively solve real-world classification problems in production environments. Our contribution is as follows: 1) we introduced a differentiable KNN layer (Xie et al., 2020) into the model architecture, 2) we proposed a novel approach to aggregate classifier predictions in the ensemble, 3) we performed exhaustive experiments showing the ability to learn incrementally and real-world usability, 4) we demonstrate the effectiveness of the proposed architecture by achieving SOTA results on various data sets without a memory buffer.

2 Related work

2.1 Continual Learning

2.1.1 Methods

Currently, methods with a memory buffer such as GEM (Lopez-Paz and Ranzato, 2017), A-GEM (Chaudhry et al., 2019a) or DER (Buzzega et al., 2020) usually achieve the highest performance in all continual learning scenarios (Mai et al., 2022). Such methods store part of the data in the memory and this data is successively replayed during training on new, unseen examples. However, the requirement to store data in memory disqualifies these methods in many practical applications due to privacy policies or data size (Salem et al., 2018). This forces attention toward other approaches, such as parameter regularization. The most popular methods in this group include EWC (Kirkpatrick et al., 2016) and LWF (Li and Hoiem, 2017). When receiving a new dose of knowledge, these methods attempt to influence the model parameter updating procedure to be minimally invasive. As research shows (Van de Ven and Tolias, 2019), regularization-based methods fail in class incremental scenarios making them ineffective in many real-world cases.

2.1.2 Approaches for NLP

Almost all prior works focus on the development of continual learning methods in the computer vision domain (Delange et al., 2021). Research on continual learning for NLP is limited and, as Biesialska et al. (2020) observed, the majority of current NLP methods are task-specific. Moreover, these methods often use a memory buffer (de Masson D'Autume et al., 2019) or relate to the language model itself (Ke et al., 2021). To address this niche, domain-agnostic approaches have to become much more prevalent in the near future.

2.2 Ensemble methods

Ensemble methods are widespread in the world of machine learning (Zhang and Ma, 2012). By using predictions of multiple weak learners, it is possible to get a model that performs surprisingly well overall. Broad adoption of methods (Cao et al., 2020; Li and Pan, 2022; Yang et al., 2021) demonstrates the effectiveness of ensemble techniques in a wide variety of tasks. Ensembles have also been used successfully in the field of continual learning, as evidenced by the BatchEnsemble (Wen et al., 2020) or CN-DPM (Lee et al., 2020). Other contributions present in literature (Doan et al., 2022) tend to focus strongly on improving model performance rather than increasing model efficiency. Furthermore, ensemble approaches can also be used indirectly through dropout (Srivastava et al., 2014) or weights aggregation (Wortsman et al., 2022).

2.3 Mixture of Experts

Mixture of Experts (ME) (Jacobs et al., 1991) is a technique based on the divide and conquer paradigm. It assumes dividing the problem space between several specialized models (experts). Experts are supervised by the gating network that selects them based on the defined strategy. The difference between the ensembles is that ME methods focus on selecting a few experts rather than combining predictions of all available models. ME techniques have found many applications in various domains (Masoudnia and Ebrahimpour, 2014), including continual learning (Shanahan et al., 2021), and even nowadays such approaches are widely used in NLP (Gao et al., 2022; Ravaut et al., 2022).

2.4 Real-world NLP systems

Over the last few years, the amount of real-world NLP applications has grown rapidly (Sarker, 2022). Despite major successes in the real-world application of language technologies such as Google Translate, Amazon Alexa, and ChatGPT, production deployment and maintenance of such models still remain a challenge. Researchers have shown (Nowakowski et al., 2022; Karakanta et al., 2021), that there are several issues related to maintaining NLP models, including technical limitations, latency, and performance evaluation. However, the crucial problem is the shift of data domain that forces models to be retrained and deployed again over time (Hu et al., 2020). It is a major limitation in dynamically changing environments where users

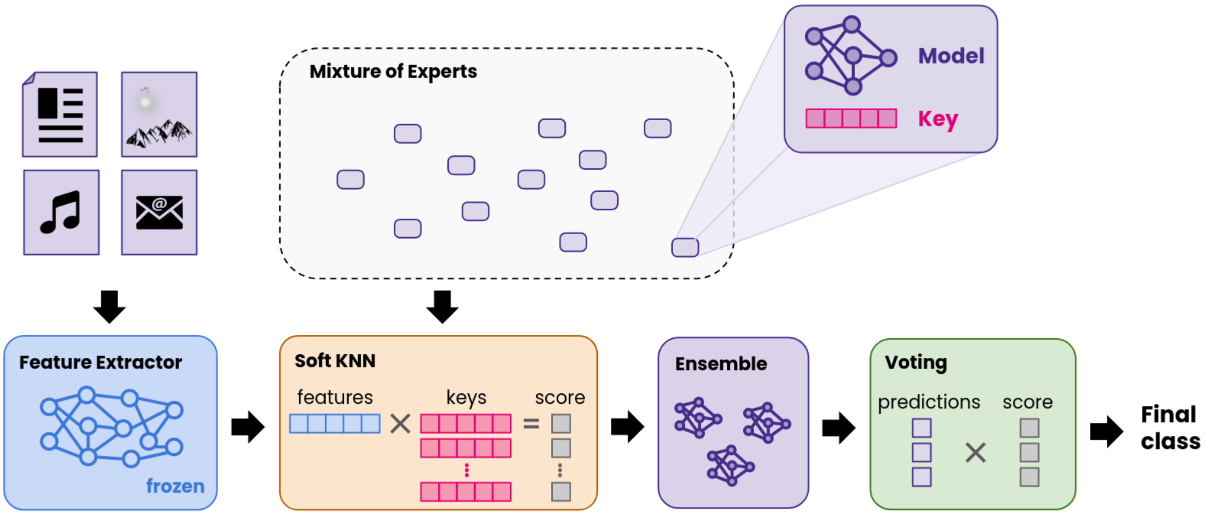

Figure 2: Architecture of the proposed model. An input is processed by the feature extractor. Obtained embeddings are used to find the most relevant classifiers according to assigned keys. The *soft KNN* layer approximates the *soft KNN* scores. Predictions are weighted in the voting layer by both cosine similarity and *soft KNN* scores. Final output is the class with the highest voting score.

expect models to quickly adapt to them. Currently, this problem has been tackled in several systems (Afzal et al., 2019; Hancock et al., 2019), but many of the solutions preclude maintaining model accuracy when training incrementally making them insufficient.

3 Our approach

3.1 Problem formulation

Class incremental continual learning involves training a classification model $f(\cdot) : \mathbb{X} \longmapsto \mathbb{Y}$ on a sequence of T tasks. The model is trained on each task separately (one task at a time). Each task D_t contains data points $D_t = \{(x_t^1, y_t^1), \ldots, (x_t^{N_t}, y_t^{N_t})\}$, where N_t is length of D_t, $x_t^{(i)} \in \mathbb{R}^D$, and $y_t^{(i)} \in \mathbb{Y}_t$. $\mathbb{Y}_t$ is a label set for task t and $\mathbb{Y}_t \cap \mathbb{Y}_{t'} = \emptyset$ for $t \neq t'$. We want the model to keep performing well on all previous tasks after each update, and we assume to be working in the most challenging setup (Van de Ven and Tolias, 2019), where one task consists of data from one class.

3.2 Method

We present a flexible and effective domain-agnostic architecture that can be used to solve various classification problems. The architecture is presented in Figure 2.

Feature extractor. The first component of the proposed architecture is a multi-layer feature extractor that transforms input data into the embedding space. It can be described by the following mapping $\mathbf{z} = F(\mathbf{x})$, where $\mathbf{x} \in \mathbb{R}^D$ is an input example and $\mathbf{z} \in \mathbb{R}^M$ is a M-dimensional embedding. The approach we follow assumes the use of a pre-trained model with frozen parameters. Such a procedure makes it possible to completely prevent the extractor from forgetting knowledge by isolating feature space learning from the classification process.

Keys and classifiers. We use an ensemble of N classifiers $f_n(\cdot)$, where each of them maps the embedding into a K-dimensional output vector $\hat{\mathbf{y}}_n = f_n(\mathbf{z})$. With each classifier, there is an associated key vector $\mathbf{k}_n \in \mathbb{R}^M$ with the same dimensionality as the embedding. The keys help to select the most suitable models for specialization with respect to the currently processed input example. They are initialized randomly from normal distribution. We use simple single-layer neural networks as classifiers, with fan-in variance scaling as the weight initialization strategy. The network output is activated by a hyperbolic tangent function (*tanh*).

Soft κ-nearest neighbors layer. The standard KNN algorithm is often implemented using ordinary sorting operations that make it impossible to determine the partial derivatives with respect to the input. It removes the ability to use KNN as part of end-to-end neural models. However, it is possible to obtain a differentiable approximation of

Table 1: Data sets setup for experiments.

Domain	Data set	Classes	Train	Test	Avg. words
Text	BBC News	5	1,668	557	380
	Newsgroups	10	11314	7532	315
	Complaints	10	16,000	4,000	228
Audio	Speech Commands	10	18,538	2,567	—
Image	MNIST	10	60,000	10,000	—
	CIFAR-10	10	50,000	10,000	—

the KNN model by solving the Optimal Transport Problem (Peyré et al., 2019). Based on this concept, we add a differentiable layer to the model architecture. We call this layer soft κ-nearest neighbors (*soft KNN*). In order to determine the KNN approximation, we first compute a cosine distance vector $\mathbf{c} \in \mathbb{R}^N$ between the embedding and the keys:

$$c_n = 1 - \cos(\mathbf{z}, \mathbf{k}_n), \tag{1}$$

where $\cos(\cdot, \cdot)$ denotes the cosine similarity. Next, we follow the idea of a soft top-κ operator presented in (Xie et al., 2020), where κ denotes the number of nearest neighbors. Let $\mathbf{E} \in \mathbb{R}^{N\times 2}$ be the Euclidean distance matrix with the following elements:

$$e_{n,0} = (c_n)^2, \quad e_{n,1} = (c_n - 1)^2. \tag{2}$$

And let $\mathbf{G} \in \mathbb{R}^{N\times 2}$ denote the similarity matrix obtained by applying the Gaussian kernel to $\mathbf{E}$:

$$\mathbf{G} = \exp(-\mathbf{E}/\sigma), \tag{3}$$

where σ denotes the kernel width. The $\exp$ operators are applied elementwise to the matrix $\mathbf{E}$.

We then use the Bregman method, an algorithm designed to solve convex constraint optimization problems, to compute L iterations of Bregman projections in order to approximate their stationary points:

$$\mathbf{p}^{(l+1)} = \frac{\boldsymbol{\mu}}{\mathbf{G}\mathbf{q}^{(l)}}, \quad \mathbf{q}^{(l+1)} = \frac{\boldsymbol{\nu}}{\mathbf{G}^\top \mathbf{p}^{(l+1)}}, \tag{4}$$

where $l = 0, \ldots, L-1, \boldsymbol{\mu} = \mathbf{1}_N/N$, $\boldsymbol{\nu} = [\kappa/N, (N-\kappa)/N]^\top$, $\mathbf{q}^{(0)} = \mathbf{1}_2/2$, and $\mathbf{1}_i$ denotes the i-element all-ones vector. Finally, let $\boldsymbol{\Gamma}$ denotes the optimal transport plan matrix and is given by:

$$\boldsymbol{\Gamma} = \text{diag}(\mathbf{p}^{(L)}) \cdot \mathbf{G} \cdot \text{diag}(\mathbf{q}^{(L)}) \tag{5}$$

As the final result $\boldsymbol{\gamma} \in \mathbb{R}^N$ of the soft κ-nearest neighbor operator, we take the second column of $\boldsymbol{\Gamma}$ multiplied by N i.e. $\boldsymbol{\gamma} = N\boldsymbol{\Gamma}_{:,2}$. $\boldsymbol{\gamma}$ is a soft approximation of a zero-one vector that indicates which κ out of N instances are the nearest neighbors. Introducing the *soft KNN* enables to train parts of the model that were frozen until now.

Voting layer. We use both c_n and $\boldsymbol{\gamma}$ to weight the predictions by giving the higher impact for classifiers with keys similar to extracted features. The obtained approximation $\boldsymbol{\gamma}$ has two main functionalities. It eliminates the predictions from classifiers outside κ nearest neighbors and weights the result. Since the Bregman method does not always completely converge, the vector κ contains continuous values that are close to 1 for the most relevant classifiers. We make use of this property during the ensemble voting procedure. The higher the κ value for a single classifier, the higher its contribution toward the final ensemble decision. The final prediction is obtained as follows:

$$\hat{\mathbf{y}} = \frac{\sum_{n=1}^{N} \gamma_n c_n \hat{\mathbf{y}}_n}{\sum_{n=1}^{N} c_n} \tag{6}$$

Training To effectively optimize the model parameters, we follow the training procedure presented in (Shanahan et al., 2021). It assumes the use of a specific loss function that is the inner product between the ensemble prediction and the one-hot coded label:

$$\mathcal{L}(\mathbf{y}, \hat{\mathbf{y}}) = -\mathbf{y}^\top \hat{\mathbf{y}} \tag{7}$$

Optimizing this criterion yields an advantage of using a *tanh* activation function, significantly reducing catastrophic forgetting (Shanahan et al., 2021). Following the reference method, we also use an optimizer that discards the value of the gradient and uses only its sign to determine the update direction. As a result, the parameters are being changed by a fixed step during the training.

4 Experiments

4.1 Setup

In order to ensure experiment's reproductivity, we evaluated our method on the popular and publicly available data sets.

Data sets We use three common text classification data sets with different characteristics - Newsgroups (Lang, 2008), BBC News (Greene and Cunningham, 2006), and Consumer Finance Complaints[2]. The goal of the experiments was to evaluate our method on tasks with with different dif-

[2] Source: https://huggingface.co/datasets/consumer-finance-complaints

Table 2: Accuracy (%) and standard deviation for methods evaluated on various data sets. Speech Commands data set was evaluated with 64 classifiers in ME, the remaining models have 128 classifiers. Regularization-based methods completely failed on the difficult data sets due to the recency bias phenomenon (Mai et al., 2022).

		Text			Image		Audio
Model	**Mem.**	**NG**	**BBC**	**Compl.**	**MNIST**	**CIFAR-10**	**Sp. Comm.**
Naive	×	5.25$_{\pm 0.03}$	21.65$_{\pm 2.56}$	9.56$_{\pm 0.33}$	11.29$_{\pm 3.05}$	10.00$_{\pm 0.01}$	21.54$_{\pm 3.78}$
LwF	×	5.20$_{\pm 0.05}$	18.60$_{\pm 2.03}$	10.04$_{\pm 0.20}$	11.47$_{\pm 2.75}$	10.00$_{\pm 0.01}$	20.61$_{\pm 3.88}$
EWC	×	5.13$_{\pm 0.13}$	21.97$_{\pm 2.14}$	10.16$_{\pm 0.31}$	11.19$_{\pm 2.70}$	10.00$_{\pm 0.01}$	32.93$_{\pm 4.92}$
SI	×	5.27$_{\pm 0.01}$	19.43$_{\pm 2.96}$	10.00$_{\pm 0.62}$	14.90$_{\pm 6.52}$	10.00$_{\pm 0.01}$	9.99$_{\pm 0.27}$
CWR*	×	4.63$_{\pm 0.60}$	22.98$_{\pm 1.20}$	10.13$_{\pm 0.33}$	10.40$_{\pm 0.54}$	10.00$_{\pm 0.01}$	10.32$_{\pm 0.26}$
GEM	✓	35.89$_{\pm 3.80}$	70.99$_{\pm 7.68}$	33.74$_{\pm 2.50}$	52.27$_{\pm 5.20}$	23.40$_{\pm 2.71}$	21.01$_{\pm 2.06}$
A-GEM	✓	9.44$_{\pm 7.14}$	59.10$_{\pm 17.52}$	9.20$_{\pm 0.01}$	65.37$_{\pm 4.53}$	26.43$_{\pm 5.27}$	17.45$_{\pm 6.90}$
Replay	✓	22.45$_{\pm 3.09}$	59.61$_{\pm 3.17}$	16.46$_{\pm 4.62}$	69.02$_{\pm 4.90}$	32.93$_{\pm 4.56}$	12.23$_{\pm 1.28}$
E&E	×	46.07$_{\pm 2.91}$	75.87$_{\pm 3.88}$	44.80$_{\pm 1.62}$	87.10$_{\pm 0.21}$	53.97$_{\pm 1.31}$	79.15$_{\pm 0.60}$
Ours	×	**47.27**$_{\pm 3.63}$	**78.49**$_{\pm 3.92}$	**44.97**$_{\pm 0.86}$	**87.62**$_{\pm 0.14}$	**56.27**$_{\pm 1.21}$	**80.11**$_{\pm 1.30}$

ficulty levels. We also conducted experiments for audio classification using Speech Commands (Warden, 2018) data set. For the evaluation purposes, we selected the 10 most representative classes from the Newsgroups, Complaints and Speech Commands. Finally, we also conducted experiments on the popular MNIST and CIFAR-10 data sets as image domain representatives. The data set summary is presented in Table 1. In all experiments we used a train set to train model incrementally, and afterward we performed a standard evaluation using a test set.

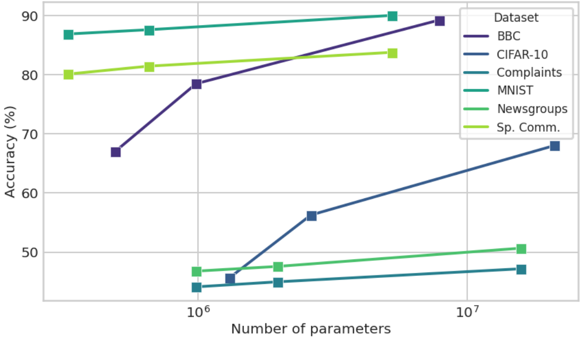

Figure 3: Number of parameters in DE&E architecture (64, 128, 1024 classifiers) and achieved accuracy (%). We calculated the number of parameters as the sum of the parameters for all classifiers in the ME. Each mark is the test accuracy averaged across 5 runs.

Feature extractors For all text data sets, we used a Distilbert (Sanh et al., 2019), a light but still very effective alternative for large language models. Next, for Speech Commands, we utilized Pyannote (Bredin et al., 2020), a pretrained model for producing meaningful audio features. For image data sets, we used different extractors. MNIST features were produced by the pretrained VAE and CIFAR-10 has a dedicated BYOL model (see A.4 for more details).

4.2 Results

The results of the evaluation are presented in Table 2. For all setups evaluated, our model performed best improving results of the main reference method (E&E) by up to 3 percent points (pp.). The improvement scale varies across the data sets. We also observed a significant difference in achieved accuracy between the DE&E and the standard continual learning methods. Simple regularization-based methods completely fail in the class incremental scenario. It shows how demanding training the model incrementally is when a set of classes is not fixed, which often takes place in real-world scenarios. Furthermore, our method achieved these results without replaying training examples seen in the past, making it more practical relative to the SOTA memory-based methods (GEM, A-GEM, Replay) that store samples from every class. For the ensemble of 128 classifiers and Speech Commands data set, our architecture achieved an accuracy of more than 59 pp. higher than the best method with a memory buffer.

One of the most important hyperparameters of the model is the number of classifiers (experts). To investigate how it affects accuracy, we evaluated our architecture in three variants: small - 64, normal - 128, and large - 1024 classifiers. The evaluation results are presented in Figure 3. We observed that increasing the ensemble size trans-

Table 3: Accuracy (%) and standard deviation of DE&E evaluated on Class Incremental and Domain Incremental scenarios. We used the same setup as shown in Table 2.

Data set	Class Incremental	Domain incremental
BBC News	$78.49_{\pm 3.92}$	$79.71_{\pm 3.14}$
Newsgroups	$47.27_{\pm 3.63}$	$44.55_{\pm 1.40}$
Complaints	$44.97_{\pm 0.86}$	$39.23_{\pm 3.03}$
Speech Commands	$81.46_{\pm 0.85}$	$79.31_{\pm 0.49}$
MNIST	$87.62_{\pm 0.14}$	$85.04_{\pm 0.39}$
CIFAR-10	$56.27_{\pm 1.21}$	$55.66_{\pm 1.32}$

lates to higher accuracy, and gain depends on the setup and data characteristics. The most significant improvement was observed on BBC and CIFAR-10 where the large model achieved an accuracy of about 20pp. better than the small one. For the remaining data sets and the analogous setup, the gain was up to 5pp. We explain this phenomenon as the effect of insufficient specialization level achieved by smaller ensembles. If experts are forced to solve tasks that are too complicated they make mistakes often. Increasing the number of experts allows for dividing feature space into simpler sub-tasks. However, such a procedure has natural limitations related to the feature extractor. If features have low quality, increasing the number of experts will be ineffective. To select the optimal ensemble size we suggest using the elbow rule which prevents the model from being overparameterized and ensures reasonable accuracy. However, in general, we recommend choosing larger ensembles that are better suited for handling real-world cases.

Since real-world environments require deployed models to quickly adapt to domain shifts, we tested our method in a domain incremental scenario. In such setup, each data batch can provide examples from multiple classes that can be either known or new (Van de Ven and Tolias, 2019). This way, the model needs to learn incrementally, being prone to frequent domain shifts. As shown in Table 3, the proposed method handles both scenarios with comparable accuracy. We observed improved accuracy for BBC News, but reduced for the remaining data sets. Such property can be beneficial when there is limited prior knowledge about the data or the stream is imbalanced (Aguiar et al., 2022).

We have also investigated the importance of the presented expert selection method. We trained the DE&E method and for each training example, we allowed it to choose random experts (rather than the most relevant ones) with fixed probability p. As shown in Figure 4, the selection method has a strong influence on the model performance. Accuracy decreases proportionally to the p over all data sets studied. The proper expert selection technique is crucial for the presented method. It is worth noting that relatively easier data sets suffer less from loss of accuracy than hard ones because even randomly selected experts can still classify the data by learning simple general patterns. In more difficult cases like Newsgroups and Complaints data sets, model performance is comparable to random guessing when $p > 0.5$.

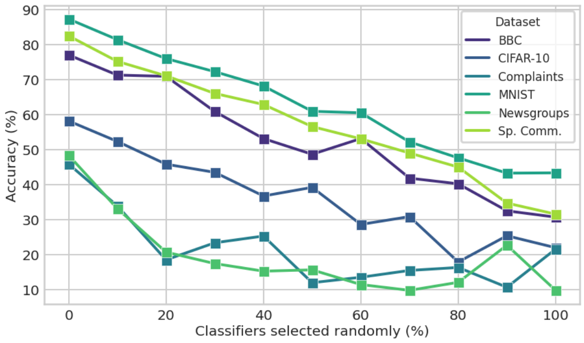

Figure 4: Influence of random classifier selection on DE&E accuracy (%). All models consist of 128 classifiers. Each mark is the accuracy for an independent run.

5 Conclusions

In this paper, we proposed a domain-agnostic architecture for continual learning with a training procedure specialized in challenging class incremental problems. The presented architecture is based on the Mixture of Experts technique and handles many practical issues related to the deployment of text classification models in non-trivial real-world systems. As our main contribution, we introduced a fully differentiable *soft KNN* layer and a novel prediction weighting strategy. By conducting exhaustive experiments, we showed improvement in accuracy for all the cases studied and achieved SOTA results without using a memory buffer. This enables an effective and secure training, especially when working with sensitive textual data. The presented architecture is highly flexible, can effectively solve classification problems in many domains, and can be applied to real-world machine learning systems requiring continuous improvement. Such work enables researchers to make further steps toward overrunning many of the current challenges related to language technology applications.

Limitations

The main limitations of the proposed architecture are related to the presence of the frozen feature extractor. The accuracy of the classification module is proportional to the quality of features. Since the ensemble weak learners are single-layer neural networks, the entire feature extraction process relies on a pre-trained model that strongly limits the upper bound of classification accuracy. Such approach reduces the method complexity, but also makes it prone to errors when embeddings have low quality. Achieving accuracy at a satisfactory level, which is crucial in real world systems, requires the use of high quality feature extractors. Currently, plenty of pretrained SOTA models are available for free in domains such as text or image classification, but if such extractor is not available, does not produce reasonable features or is too expensive to use, our architecture may not be the best choice.

Another issue is relatively long training time comparing to the reference methods (see A.3). The introduction of a differentiable *soft KNN* layer resulted in additional computational effort that clearly impacted the model complexity. This limits the use in low latency systems with machine learning models trained online.

Ethics Statement

The authors foresee no ethical concerns with the work presented in this paper, in particular concerning any kind of harm and discrimination. Since the presented architecture can have a wide range of usages, the authors are not responsible for any unethical applications of this work.

Acknowledgements

The research was conducted under the Implementation Doctorate programme of Polish Ministry of Science and Higher Education and also partially funded by Department of Artificial Intelligence, Wroclaw Tech and by the European Union under the Horizon Europe grant OMINO (grant number 101086321). It was also partially co-funded by the European Regional Development Fund within the Priority Axis 1 "Enterprises and innovation", Measure 1.2. "Innovative enterprises, sub-measure 1.2.1. "Innovative enterprises – horizontal competition" as part of ROP WD 2014-2020, support contract no. RPDS.01.02.01-02-0063/20-00.

References

Shazia Afzal, Tejas Dhamecha, Nirmal Mukhi, Renuka Sindhgatta, Smit Marvaniya, Matthew Ventura, and Jessica Yarbro. 2019. Development and deployment of a large-scale dialog-based intelligent tutoring system. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Industry Papers)*, pages 114–121, Minneapolis, Minnesota. Association for Computational Linguistics.

Gabriel Aguiar, Bartosz Krawczyk, and Alberto Cano. 2022. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework. *arXiv preprint arXiv:2204.03719*.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R Costa-Jussa. 2020. Continual lifelong learning in natural language processing: A survey. *arXiv preprint arXiv:2012.09823*.

Hervé Bredin, Ruiqing Yin, Juan Manuel Coria, Gregory Gelly, Pavel Korshunov, Marvin Lavechin, Diego Fustes, Hadrien Titeux, Wassim Bouaziz, and Marie-Philippe Gill. 2020. pyannote.audio: neural building blocks for speaker diarization. In *ICASSP 2020, IEEE International Conference on Acoustics, Speech, and Signal Processing*, Barcelona, Spain.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. 2020. Dark experience for general continual learning: a strong, simple baseline. *Advances in neural information processing systems*, 33:15920–15930.

Yue Cao, Thomas Andrew Geddes, Jean Yee Hwa Yang, and Pengyi Yang. 2020. Ensemble deep learning in bioinformatics. *Nature Machine Intelligence*, 2(9):500–508.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. 2019a. Efficient lifelong learning with a-gem. In *ICLR*.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019b. Continual learning with tiny episodic memories. *arXiv preprint arXiv:1902.10486, 2019*.

Cyprien de Masson D'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning. *Advances in Neural Information Processing Systems*, 32.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Thang Doan, Seyed Iman Mirzadeh, Joelle Pineau, and Mehrdad Farajtabar. 2022. Efficient continual learning ensembles in neural network subspaces. *arXiv preprint arXiv:2202.09826*.

Robert M French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3(4):128–135.

Ze-Feng Gao, Peiyu Liu, Wayne Xin Zhao, Zhong-Yi Lu, and Ji-Rong Wen. 2022. Parameter-efficient mixture-of-experts architecture for pre-trained language models. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3263–3273, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Derek Greene and Pádraig Cunningham. 2006. Practical solutions to the problem of diagonal dominance in kernel document clustering. In *Proc. 23rd International Conference on Machine learning (ICML'06)*, pages 377–384. ACM Press.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3667–3684, Florence, Italy. Association for Computational Linguistics.

Yipeng Hu, Joseph Jacob, Geoffrey JM Parker, David J Hawkes, John R Hurst, and Danail Stoyanov. 2020. The challenges of deploying artificial intelligence models in a rapidly evolving pandemic. *Nature Machine Intelligence*, 2(6):298–300.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87.

Alina Karakanta, Sara Papi, Matteo Negri, and Marco Turchi. 2021. Simultaneous speech translation for live subtitling: from delay to display. In *Proceedings of the 1st Workshop on Automatic Spoken Language Translation in Real-World Settings (ASLTRW)*, pages 35–48, Virtual. Association for Machine Translation in the Americas.

Zixuan Ke, Bing Liu, Nianzu Ma, Hu Xu, and Lei Shu. 2021. Achieving forgetting prevention and knowledge transfer in continual learning. *Advances in Neural Information Processing Systems*, 34:22443–22456.

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. 2016. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796.

Lang. 2008. 20 newsgroups dataset.

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. 2020. A neural dirichlet process mixture model for task-free continual learning. *CoRR*, abs/2001.00689.

Yang Li and Yi Pan. 2022. A novel ensemble deep learning model for stock prediction based on stock prices and news. *International Journal of Data Science and Analytics*, 13(2):139–149.

Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence*, 40(12):2935–2947.

Vincenzo Lomonaco and Davide Maltoni. 2017. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pages 17–26. PMLR.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6470–6479, Red Hook, NY, USA. Curran Associates Inc.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. 2022. Online continual learning in image classification: An empirical survey. *Neurocomputing*, 469:28–51.

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. *The Artificial Intelligence Review*, 42(2):275.

Artur Nowakowski, Krzysztof Jassem, Maciej Lison, Rafał Jaworski, Tomasz Dwojak, Karolina Wiater, and Olga Posesor. 2022. nEYron: Implementation and deployment of an MT system for a large audit & consulting corporation. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 183–189, Ghent, Belgium. European Association for Machine Translation.

Gabriel Peyré, Marco Cuturi, et al. 2019. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607.

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. 2022. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524, Dublin, Ireland. Association for Computational Linguistics.

Waseem Rawat and Zenghui Wang. 2017. Deep convolutional neural networks for image classification: A comprehensive review. *Neural computation*, 29(9):2352–2449.

Ahmed Salem, Yang Zhang, Mathias Humbert, Mario Fritz, and Michael Backes. 2018. Ml-leaks: Model and data independent membership inference attacks and defenses on machine learning models. *CoRR*, abs/1806.01246.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Iqbal H Sarker. 2022. Ai-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems. *SN Computer Science*, 3(2):158.

Murray Shanahan, Christos Kaplanis, and Jovana Mitrovic. 2021. Encoders and ensembles for task-free continual learning. *CoRR*, abs/2105.13327.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Gido M Van de Ven and Andreas S Tolias. 2019. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Pete Warden. 2018. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*.

Yeming Wen, Dustin Tran, and Jimmy Ba. 2020. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *arXiv preprint arXiv:2203.05482*.

Yujia Xie, Hanjun Dai, Minshuo Chen, Bo Dai, Tuo Zhao, Hongyuan Zha, Wei Wei, and Tomas Pfister. 2020. Differentiable top-k with optimal transport. In *Advances in Neural Information Processing Systems*, volume 33, pages 20520–20531. Curran Associates, Inc.

Yongquan Yang, Haijun Lv, and Ning Chen. 2021. A survey on ensemble learning under the era of deep learning. *arXiv preprint arXiv:2101.08387*.

Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR.

Cha Zhang and Yunqian Ma. 2012. *Ensemble machine learning: methods and applications*. Springer.

A Appendix

A.1 Code

Code is currently available as a Github repository `https://github.com/mateusz-wojcik-97/domain-agnostic-architecture`.

A.2 Computing resources

The machine we used had 128 GB RAM, an Intel Core i9-11900 CPU, and an NVIDIA GeForce RTX 3060 GPU with 12GB VRAM. Every experiment was performed using the GPU.

A.3 Time complexity

Table 4: Time (seconds) of training the ensemble models with 128 classifiers on one task.

Dataset	E&E	Ours
Newsgroups	7.43	31.20
BBC News	14.96	151.79
Complaints	20.33	93.63
Sp. Commands	30.80	108.90
MNIST	28.01	270.30
CIFAR-10	104.25	355.82

The comparison in training time between E&E and DE&E models is shown in Table 4. For all evaluated data sets, the training time of our model was higher than the time to train the reference method. The results vary between data sets. The introduction of a differentiable *soft KNN* layer resulted in additional computational effort that clearly impacted the time complexity of the model.

A.4 Implementation details

We use PyTorch to both reproduce the E&E results and implement the DE&E method. For text classification we used pretrained Distilbert [3] model and for audio classification we used pretrained Pyannote [4] model, both from the Huggingface repository. We used a pre-trained ResNet-50 model as the feature extractor for the CIFAR-10 data set. The model is available in the following GitHub repository,

[3] https://huggingface.co/distilbert-base-uncased

[4] https://huggingface.co/pyannote/embedding

https://github.com/yaox12/BYOL-PyTorch, and is used under MIT Licence. For MNIST, we trained a variational autoencoder on the Omniglot data set and utilized encoder part as our feature extractor. We based our implementation of the *soft KNN* layer on the code provided with https://proceedings.neurips.cc/paper/2020/hash/ec24a54d62ce57ba93a531b460fa8d18-Abstract.html. All data sets used are public.

Table 5: Architecture of neural networks used as backbones for baseline models depends on experimental setup. Each network has a similar number of total parameters as in the ensemble.

Dataset	Network layers
Newsgroups	[1536, 1700, 768, 10]
Complaints	[1536, 955, 512, 10]
BBC News	[1536, 640, 5]
Sp. Commands	[512, 1256, 10]
MNIST	[512, 1256, 10]
CIFAR-10	[2048, 1274, 10]

Baselines We use Naive, LwF (Li and Hoiem, 2017), EWC (Kirkpatrick et al., 2016), SI (Zenke et al., 2017), CWR* (Lomonaco and Maltoni, 2017), GEM (Lopez-Paz and Ranzato, 2017), A-GEM (Chaudhry et al., 2019a) and Replay (Chaudhry et al., 2019b) approaches as baselines to compare with our method. We utilize the implementation from Avalanche (https://avalanche.continualai.org/), a library designed for continual learning tasks. The main purpose of this comparison was to determine how the proposed method performs against classical approaches and, in particular, against the methods with memory buffer, which gives a significant advantage in class incremental problems. The recommended hyperparameters for each baseline method vary across usages in literature, so we chose them based on our own internal experiments. For a clarity, we keep hyperparameter naming nomenclature from the Avalnache library. For EWC we use $lambda$ = 10000. The LwF model was trained with $alpha$ = 0.15 and $temperature$ = 1.5. For SI strategy, we use $lambda$ = $5e7$ and eps = $1e - 7$. The hyperparameters of the memory based approach GEM were set as follows: $memory\_strength$ = 0.5, $patterns\_per\_exp$ = 5, which implies that with every task, 5 examples will be accumulated. This has a particular importance when the number of classes is large. With this setup and 10 classes in data set, memory contains 50 examples after training on all tasks. Having a large memory buffer makes achieving high accuracy much easier. For the A-GEM method, use the same number of examples in memory and $sample\_size$ = 20. All models were trained using Adam optimizer with a $learning\_rate$ of 0.0005 and $batch\_size$ of 60. We chose cross entropy as a loss function and performed one training epoch for each experience. To fairly compare baseline methods with ensembles, as a backbone we use neural network with a similar number of parameters (as in ensemble). Network architectures for each experimental setup are shown in Table 5. All baseline models were trained by providing embeddings produced by feature extractor as an input.

Ensembles. We used E&E (Shanahan et al., 2021) as the main reference method. It uses an architecture similar to that of a classifier ensemble, however the nearest neighbor selection mechanism itself is not a differentiable component and the weighting strategy is different. In order to reliably compare the performance, the experimental results of the reference method were fully reproduced. Both the reference method and the proposed method used exactly the same feature extractors. Thus, we ensured that the final performance is not affected by the varying quality of the extractor, but only depends on the solutions used in the model architecture and learning method.

Both E&E and our DE&E were trained with the same set of hyperparameters (excluding hyperparameters in the *soft KNN* layer for the DE&E). We use ensembles of sizes 64, 128 and 1024. Based on the data set, we used different hyperparameter sets for the ensembles (Table 6).

The keys for classifiers in ensembles were randomly chosen from the standard normal distribution and normalized using the $L2$ norm. The same normalization was applied to encoded inputs during lookup for matching keys.

Soft KNN. We use the Sinkhorn algorithm to perform the forward inference in *soft KNN*. The Sinkhorn algorithm is useful in entropy-regularized optimal transport problems thanks to its computational effort reduction. The *soft KNN* has $\mathcal{O}(n)$ complexity, making it scalable and allows us to safely apply it to more computationally expensive problems.

The values of *soft KNN* hyperparameters were

Table 6: Hyperparameters used for DE&E and E&E methods.

Dataset	Classifiers	Neighbors	Batch size	Learning rate	Weight Decay
Newsgroups	64	16	8	0.0001	0.0001
	128	32			
	1024	64			
BBC News	64	8	1	0.01	
	128	16			
	1024	32			
Complaints	64	16	8	0.0001	
	128	32			
	1024	64			
Sp. Commands	64	16	8	0.001	
	128	32			
	1024	64			
MNIST	128	16	60	0.0001	
CIFAR-10	128	16	60	0.0001	

$\sigma = 0.0005$ and $L = 400$. We utilize the continuous character of an output vector to weight the ensemble predictions. It is worth noting that we additionally set the threshold of the minimum allowed *soft KNN* score to 0.3. It means every element in γ lower than 0.3 is reduced to 0. We reject such elements because they are mostly the result of non-converged optimization and do not carry important information. In this way, we additionally secure the optimization result to be as representative as possible.