Improving the Continual Learning Model VAG

BY

GIUSEPPE GABRIELE
B.S. Politecnico di Torino, Turin, Italy, 2022

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2024

Chicago, Illinois

ACKNOWLEDGMENTS

There are many people I would like to thank, and I will probably forget someone. I would like to thank my advisor, Prof. Liu, and the Italian professors (Prof. Di Carlo and Prof. Savino) who helped me during the progress of this thesis. I also want to thank all my friends and all the wonderful people I met during my time in Chicago over the course of almost a year: Davi, Simo, Giò, Beppe, Nico, Falco, Lore, Alice, Marti, Tankerflat, and many others. Of course, I can't forget about my roommates Fede, Franci, and MattiChiurlo.

I want to send my regards to my lifelong friends from Sersale: Cristiano, Domenico, Tommaso, Ciccio Jojo, Peppe Frustaci, Ciccio Borelli, Lucio, GP9 and l'Ottavo as well as all the other friends from my small city and its surroundings, the marinoti and gli Statali, Pietro Pitirro, Antonio e Vrisci, il Ragioniere & Stefano, Lorenzo Kobe Veneziani, and Birillo. Other special regards for my friends Stefano, Riccardo, Massimo, Fabiana, Erika, Martina.

I also want to say thanks to all my friends from Turin for all the based moments passed together, hoping I do not forget anybody: Foros, Daniele, Samuele, Rosario, Foxy, Roberto, Martino, Gaia, Miriana, Danilo, Cammello, Claudia, Alessio, Angelo, Pinguino, La Vita and Gogogecco. I would like to say hi to my old roommates: Filippo, Salvo, and Antonino. I also have to thank Elisabetta for supervising and managing my brother. A special thanks is to Chiara, who came to Chicago to support me.

Other thanks go to my entire family: all my uncles - zio Pino, zio Franco, zio Angelo, zia Teresa, zia Daniela, and zia Caterina; my cousins - Francesco, Fabio, Giuseppe, Federica, Cristina and

ACKNOWLEDGMENTS (continued)

Giulia, the littles Teresa, Leone, Francesco, Andrea, Paride, Diego, my grandmother Lina, and a special thought for zio Mimmo and Nonno Peppe.

The biggest thanks are to my dad and mom, Salvatore and Teresa, who stayed with me even when we were miles apart. Last but not least, my brother Francesco, who has been an incredible support, helping me through numerous challenging times with patience and (not too much) understanding.

Ringrazio tutte le persone che mi sono state vicine durante questo percorso, con la speranza di festeggiare ancora con voi e continuare a morzelliare di tutto. Ca pue quandu scindu ne facimu na MANGIUATA.

`My brain:39°00’17.8"N 16°44’25.6"E - My heart:38°54’08.5"N 16°47’41.6"E`

GG

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

VAG	Vocabulary-Aware Label Generation
EWC	Elastic Weight Consolidation
TIL	Task-Incremental Learning
CIL	Class-Incremental Learning
CF	Catastrophic Forgetting
NLP	Natural Language Processing
PLM	Pre-Trained Language Model
LM-Head	Language Modelling Head

SUMMARY

In recent years, there has been significant progress in developing computers and machines that can think like humans. This field of study is known as Continual Learning.

One of the essential characteristics of human-like thinking is the capacity to recall past events and acquire new knowledge without forgetting what was formerly learned. This last aspect is crucial since one of the most significant challenges with continual learning is Catastrophic Forgetting. The process of learning new things can cause the model to forget the past, which is a significant issue that needs to be addressed. Various solutions exist for multi-task environments, including Vocabulary-Aware Label Generation (VAG). The VAG model will be improved using a multi-label approach with more instances for each dataset and mixing different techniques to avoid Catastrophic Forgetting, thereby increasing accuracy.

The initial attempts to improve the VAG involved modifying the class labels in the used datasets (BANKING77 and CLINC150, both public and used by the original VAG model too), but this did not yield positive results. Eventually, better results were achieved by adding labels supporting each sentence, rather than modifying them. This resulted in having more than one label for each input sentence and led to improved accuracy compared to the original VAG. A new and improved sentence transformer will be used to increase accuracy further. Finally, a combination of the VAG and the Elastic-Weight Combination (EWC) will be demonstrated, resulting in the best accuracy presented in this project.

CHAPTER 1

INTRODUCTION

Machine learning nowadays has witnessed remarkable advancements, driven by the relentless pursuit of more intelligent and adaptable algorithms. Specifically, neural networks, inspired by the parallel processing and interconnection of neurons in the human brain, have catalyzed transformative successes in artificial intelligence. These computational models, consisting of layers of interconnected nodes, excel at learning intricate patterns and relationships from vast datasets. Across diverse fields, neural networks have revolutionized problem-solving capabilities.

In the domain of Natural Language Processing (NLP), neural networks have emerged as indispensable tools, reshaping the landscape of human-computer interaction. Leveraging sophisticated algorithms, neural networks enable machines to comprehend, generate, and manipulate human language with unprecedented accuracy and nuance. Through tasks such as sentiment analysis, machine translation, and text summarization, they navigate the complexities of language, unlocking insights and applications previously unimaginable.

1.1 Continual Learning

One key area of focus within this domain is continual learning (6), a paradigm that addresses the fundamental challenge of learning and adapting to new information over time. Continual learning stands as a crucial pillar in the quest to develop machine learning systems that can effectively navigate dynamic and evolving environments.

Continual Learning is a field of study in machine learning and Natural Language Processing that faces the challenge of Catastrophic Forgetting (CF) (7), a phenomenon where a model trained on one task forgets how to perform previously learned tasks when trained on new data. To tackle this challenge, a pre-trained language model (PLM) is often utilized, plus different techniques developed in the last years to allow the model to learn and adapt to new tasks without forgetting previous ones.

There are two distinct types of learning in Continual Learning: Task-Incremental Learning (TIL) and Class-Incremental Learning (CIL) (8). In TIL, tasks are presented sequentially, and the entire dataset can be classified based on the knowledge of the task-ID. However, CIL is considered the most challenging category, as the information on the task ID is not given during the inference.

The VAG model (5) works on CIL, reducing vocabulary size and increasing dataset using augmentation. One of its main areas of interest is classification. The goal is to train the model so that, during inference, it can accurately classify each sentence with the correct label.

To improve a model's accuracy, adding one or more summary sentences to the training sets used as labels is possible. These sentences can be used to support the original label in cases where there is uncertainty about the correct label during inference. This procedure can be achieved using more instances of the same model but working with different labels for each class.

Moreover, the accuracy can be further increased if more techniques are mixed together, plus some sentence transformers can be used to capture the semantic meaning of the input text.

1.2 Background

To enable the VAG model to accommodate Class Incremental Learning, it relies on a pre-trained model: BART (Bidirectional and Auto-Regressive Transformers)(4), an extension of the transformer architecture, is adept at both bidirectional and auto-regressive tasks, making it versatile for a wide range of natural language processing applications. The BART model will be better explained in the next sections. During training, the VAG model fine-tunes the pre-trained BART model with different tasks, trying to avoid Catastrophic Forgetting.

This section will show the notions and techniques used for the VAG improvements.

1.2.1 Neural Networks

Neural networks are models that take inspiration from the human brain (they are similar to the idea of neurons). They are composed of interconnected nodes arranged in layers, with each layer processing input data using specific operations. Neural networks can learn complex patterns and relationships within data, which is why they are valuable and intensively exploited for various machine-learning tasks, such as regression and classification.

There are three main components of neural networks: input layers, hidden layers, and output layers. The input layer receives the initial data or features to be processed by the network. Each neuron in the input layer represents an individual input data feature. The hidden layer performs complex computations by applying weighted transformations to the input data. Finally, the output layer is used to retrieve the final results obtained by using the neural network. An example of the neural network is given by Figure 1. The input layer is the first layer, and it is connected to the hidden layer, which in turn is connected to the final output layer. Typically,

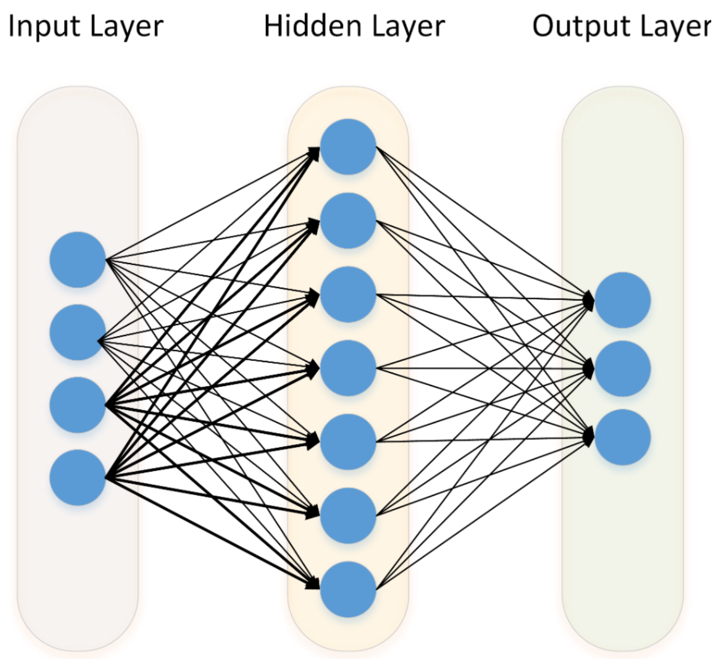


Figure 1: Schematic example of a neural network divided in its main parts, (1)

the hidden layer is composed of more than one layer. Both the case presented in the image and the one which will be used by the VAG model are a fully connected neural network, meaning each node is connected with all the nodes of the next layer.

Each layer l has a weight matrix W and bias vector b. The output h^l is calculated as:

$$h^l = f(W^l h^{l-1} + b^l), \tag{1.1}$$

where f is a pre-defined activation function and h^{l-1} is the output of the previous layer. There are different types of activation functions, shown in Figure 2. They are utilized to convert the linear output provided by the network into a non-linear result. This enables a neural network to represent more intricate functions.

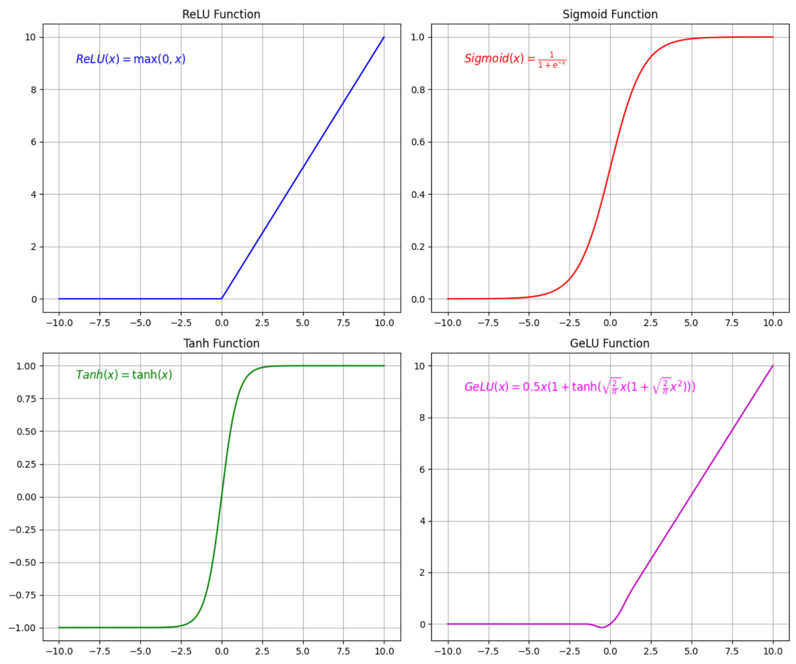


Figure 2: Different types of activation functions.

In the VAG model, a neural network is also used to create the language model head. The model is a fully connected neural network that predicts the next token for sentences.

In order to update the weights of the neural network, a loss function is always required. First, the input data is predicted, and the loss function computes the discrepancy between predicted and actual outputs. Then, the gradients of the loss function with respect to weights are computed using back-propagation. Finally, the weights are updated using the next formula:

$$W^l \leftarrow W^l - \eta \nabla L(W^l). \tag{1.2}$$

η is the learning rate, and $\eta \nabla L(W^l)$ is the gradient of the loss function with respect to the weights.

The VAG model uses the Cross-Entropy Function to measure the difference between the predicted and actual probability distributions of target labels during training.

1.2.2 Natural Language Processing

Natural Language Processing (NLP) is a sub-field of artificial intelligence that aims to teach computers how to comprehend, understand, and produce human language in a meaningful and useful manner. NLP involves several tasks, such as text analysis, language translation, sentiment analysis, and speech recognition. In the VAG model, NLP is primarily used for classification, where different techniques and components are utilized to enable the computer to "think" like a human.

The primary components involved in the VAG model are:

- Tokenization: breaking down text into smaller units, such as words or sub-words, to facilitate further analysis.

- Word Embedding: in this method, the input sentence is mapped into a continuous vector space, in order to capture the semantic and syntactical meaning of the phrase, and trying to understand the relationship between all the words of the given sentence.
- Attention Mechanism: it allows models to selectively focus on different parts of the input sequence. It will be described more clearly in one of the upcoming sub-sections, Transformers.

1.2.3 Pre-Trained Language Model

A Pre-Trained Language Model (PLM) is a machine-learning model trained using huge quantities of text data. The main goal of a PLMs is to understand and generate human-like language; that's why they are highly exploited in NLP tasks like translation, summarization, extraction, and, just like the VAG model, classification.

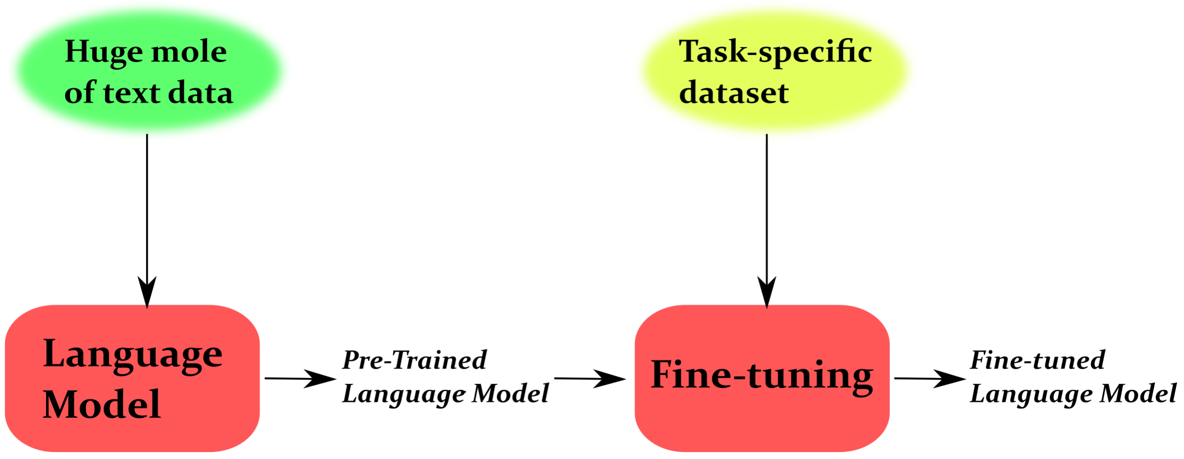


Figure 3: Figure showing how does a generic Pre-Trained Language model works

Each PLM usually consists of two main parts: pre-training and fine-tuning, as shown in Figure 3. In the pre-training phase, the language model is trained to predict the next word or token in a text sequence, using the context from the words that come before it. This method is typically known as self-supervised learning and allows the model to understand language syntax, semantics, and context. Moreover, these pre-trained models often use complex neural network structures like Transformers, which will be described in the next sections.

Fine-tuning involves training a model using a dataset specifically annotated for a particular task. This process enables the model to identify patterns relevant to the task, significantly boosting its performance.

1.2.4 Ensemble Methods

Ensemble methods (2) are advanced methods used in machine learning and deep learning (Ensemble Deep Learning) based on combining multiple different models, usually with the same architecture, to produce a better output.

The main idea is that using outputs of more than one model and then unifying them in a unique result can lead to a better accuracy w.r.t a single model.

Figure 4 shows how all the ensemble methods work: given a common input, it is passed through all the N models. Each network will produce an output o_i; then, there are different techniques to combine the outputs, and one of the most common and effective techniques is averaging all the predictions. The obtained ensemble output $\hat{o}$ will be the final ensemble model result.

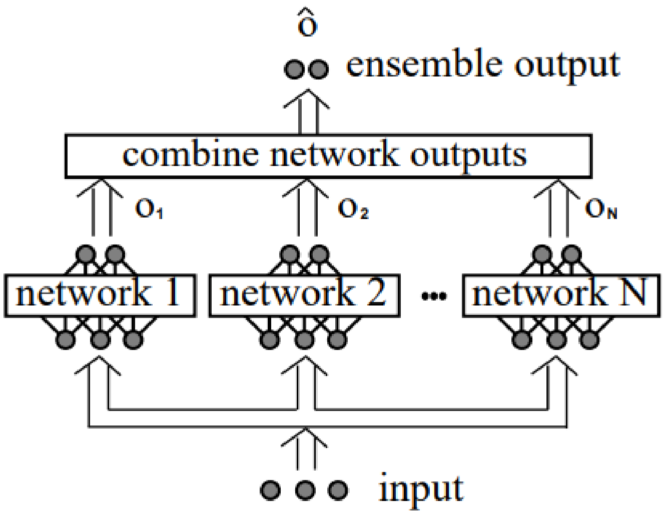


Figure 4: Figure showing the generic ensemble method technique, (2)

There are different types of ensemble methods, but they can be summed up in three different categories:

- **Bagging:** it is used for reducing the variance of the model. For each sub-model, different subsets of the training data are taken through bootstrap sampling, randomly selecting from the dataset with replacement.
- **Boosting:** it sequentially combines several models. Each model's goal is to correct the "errors" made by the past model.
- **Stacking:** it trains different models, called base learners, and combines the output using a meta-model, trained on the prediction of the base learners.

The ensemble methods are pretty similar to the idea of Multi-Label VAG since different models are used for generating a single output. However, the new model is not considered an ensemble method since it doesn't fall into any of the previous categories. In fact, as is shown in the next chapters, the combination of the outputs done during the inference part is just a summation of the final results.

1.2.5 CIL

A dataset is split into a specific number of tasks T, and each task t has many classes or labels. Given the input i of the dataset: $\{x_i, y_i\}$, the class label y_i belongs to a unique class, such that if $y_i \in t$, and $y_i \notin t'$, where $t \neq t'$. Each model learns all the tasks individually; then, the test set is used to evaluate the accuracy without knowing the task ID.

1.2.6 Transformers

Transformers (3) is a type of neural network architecture that has revolutionized various NLP tasks in recent years. These models are based on the concept of self-attention and do not rely on recurrent or convolutional structures, making them highly parallelizable and efficient. The transformer architecture consists of an encoder and decoder that use multi-head self-attention and feedforward neural networks to process input sequences.

Given an embedded input I of size (S, d_m), that are respectively the length of the sequence and the size of the embedding vector, it will be multiplied by three different weight matrices in order to produce Q (Query), K (Key) and V (Value) matrices, with the same size as the input. Each matrix is vertically separated into h parts. Given the attention formula:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{1.3}$$

where d_k is the size of V, the multi-head attention is calculated as:

$$\text{head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{1.4}$$

$$\text{multihead}(Q, K, V) = \text{concat}(\text{head}_1, ..., \text{head}_\text{h})W^O \tag{1.5}$$

During training, the input sequence is fed into the model, embedded, and used for generating Q, K, and V. This technique helps the model to understand context and dependencies inside a sentence and focusing on the relationship between the used words, capturing the meaning of the phrase, making it a popular choice for tasks such as language translation, sentiment analysis, and question-answering.

The architecture of the transformer is illustrated in Figure 5, depicting all the components within it.

The input is initially tokenized (split into small parts, such as words or groups of characters) and then transformed into a high-dimensional vector. Additionally, positional encoding is also considered to provide information about the position of each token in the sequence. Next, the encoder processes the input, which utilizes the previously described Multi-Head Attention

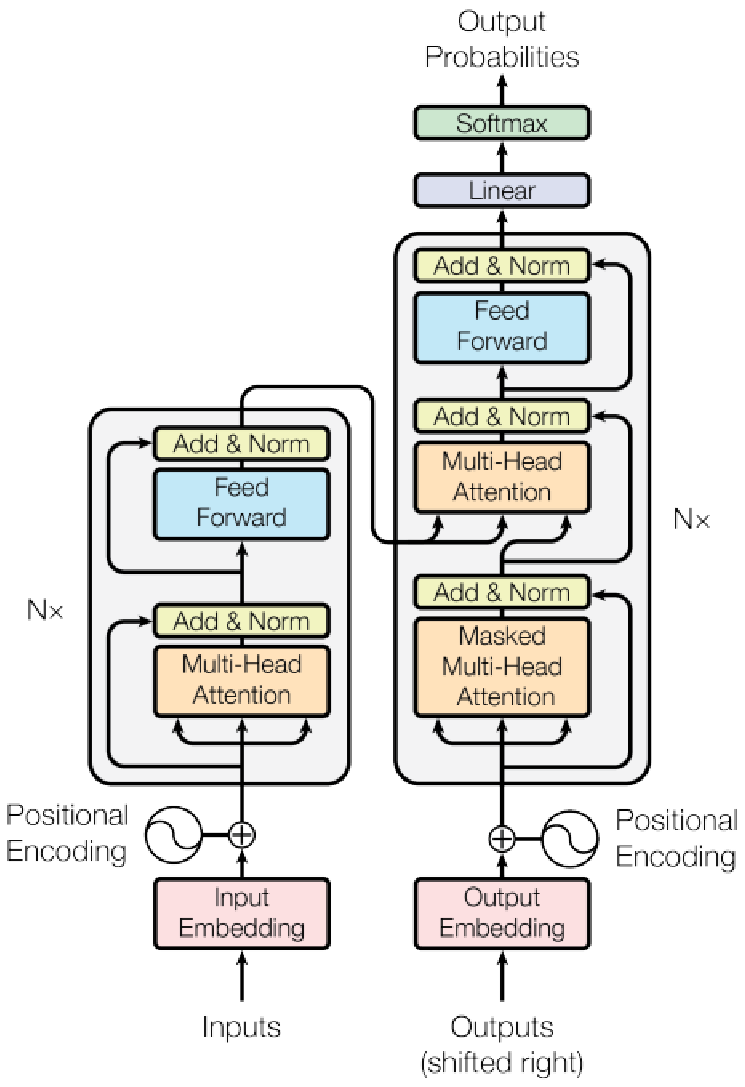


Figure 5: Complete architecture of the transformer model, (3)

mechanism. A residual connection is implemented between the input and the Multi-Head Output, followed by normalization to conclude the first part of the encoding process. The latter part of the encoder involves a feed-forward neural network, which is characterized by two linear transformations, using a ReLU function. This is followed by applying a residual connection and normalization once again. The encoder, as shown in the figure, is cycled N times.

The resulting output is then passed to the decoder, which is composed of layers made by Multi-Head attention, residual connection, and normalization, similar to the first part of the encoder. The first layer is used by the target sequence and shifted to the right so that it doesn't see the current token that should be predicted. The output of the first layer and the output of the encoder are then used for the second and third layers, where the Key K and the value V are from the encoder, and the query Q is from the target sequence. These three layers are repeated N times as well, and the output is passed to a linear connection and a softmax in order to produce the new token, which will be added to the output embedding.

This process is applied until a special ending token (like $< \text{EOS} >$, End Of Sequence) is produced.

1.2.7 BART model

BART (Bidirectional and Auto-Regressive Transformers)(4) is a sequence-to-sequence pre-training model based on the Transformer architecture, using six or twelve layers in the encoder-decoder, but instead of the ReLu function, it uses a GeLu function. It combines elements from both BERT (Bidirectional Encoder Representations from Transformers)(9) and GPT (Generative Pre-trained Transformer)(10) in order to capture bidirectional context information from the input text while generating coherent and contextually relevant output sequences.

For the pre-training model, BART is trained through the process of corrupting texts and subsequently optimizing a loss function for their reconstruction, and different types of transformations were used: Token Masking, Token Deletion, Text Infilling, Sentence Permutation, and Document Rotation.

Fine-tuning the BART model could be useful for several different tasks, such as text generation, translation, and comprehension.

The BART encoder processes the input sentence bidirectionally, while the decoder generates the output sequence in an autoregressive way.

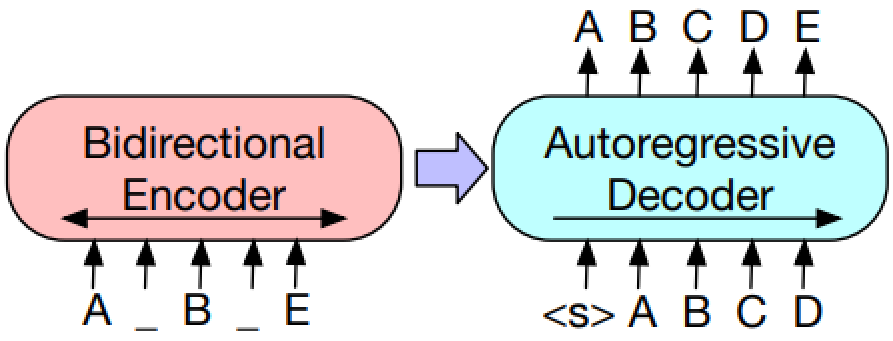


Figure 6: BART model, (4)

A graphical representation of the BART model is given in Figure 6.

Using the author's words (4), *"a document has been corrupted by replacing spans of text with mask symbols. The corrupted document (left) is encoded with a bidirectional model, and then the likelihood of the original document (right) is calculated with an autoregressive decoder"*.

During fine-tuning, an uncorrupted input is given both to the Encoder and Decoder.

CHAPTER 2

PREVIOUS WORKS

2.1 VAG

The VAG model is designed specifically for classification in CIL systems, utilizing various techniques. During the training process, a pre-trained language model, denoted by M, is fine-tuned using a specific training set, where each sentence is labeled. The model will generate an output used to calculate the loss, which is then used to fine-tune the model. To predict the best class of each sentence, a pool of all the seen labels, labeled as P, is maintained. When the input sentence is given, label y_{gen} is generated, and during the inference phase cosine-similarity is used to determine the best class:

$$y_{pred} = \underset{y \in P}{\operatorname{argmax}}(\cos(\operatorname{embed}(y), \operatorname{embed}(y_{gen}))). \tag{2.1}$$

The original *embed* function is the sentence transformer: *paraphrase-MiniLM-L6-v2* (11).

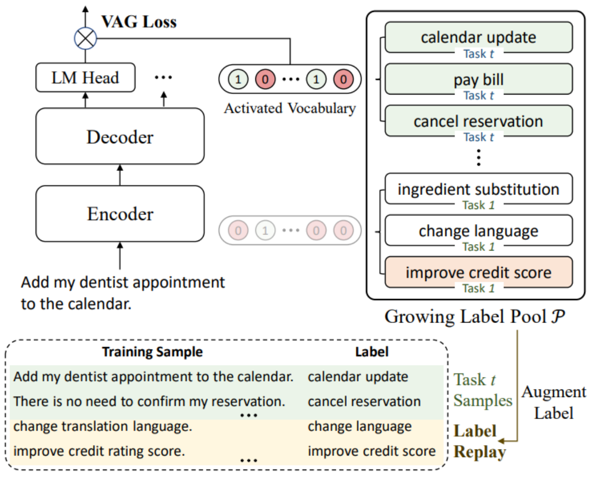
The VAG model has a distinctive feature known as the *VAG Loss.* This is due to the fact that a large number of tokens in the vocabulary are not actually used. To mitigate the negative impact of catastrophic forgetting, the VAG model masks the unused tokens. The token probability, given the embedded input z and the previously generated tokens $Y_{1:i-1}$, is:

$$P(y_i|z, Y_{1:i-1}) = \frac{exp(E_{y_i}^T f_{\theta_{\text{dec}}}(z, Y_{1:i-1}))}{\sum_{w\in\nu} exp(E_w^T f_{\theta_{dec}}(z, Y_{1:i-1}))}. \tag{2.2}$$

E_w is the word embedding of w, while ν is the vocabulary. Using a smaller ν provides a better accuracy of the model.

The VAG model applies Label-based Pseudo Replay to prevent catastrophic forgetting. This involves augmenting the dataset with additional data related to past tasks and using them to improve accuracy of each task.

VAG achieved higher accuracy than other techniques in different datasets and correctly classified labels. The project aims to improve performance starting from this model.



Training Sample	Label
Add my dentist appointment to the calendar.	calendar update
There is no need to confirm my reservation.	cancel reservation
...	
change translation language.	change language
improve credit rating score.	improve credit score
...	

Figure 7: Figure showing an overview of the original VAG model (5)

Figure 7 shows how the VAG model is trained given a task t. The probability of the next token is modified by masking the vocabulary and label-based pseudo-replay is applied for augmenting the dataset.

2.2 EWC

Elastic Weight Consolidation (EWC) (12) is another technique used in machine learning to overcome the problem of catastrophic forgetting. EWC helps the neural network to retain important knowledge from previous tasks while learning new ones. This is obtained by calculating the relevance of each weight in the neural network for a given task and constraining future learning to avoid changing those important weights too much. EWC addresses this by introducing a penalty term to the loss function during training. This penalty term encourages the network weights to stay close to the values learned during previous tasks.

Given the task-specific loss function $L_B(\theta)$, the current value of the i-th weight in the network θ_i, the value of the i-th weight after training on the previous task(s) $\theta^*_{A,i}$, the estimated variance of the i-th weight, the Fisher information matrix F_i, the new loss is calculated as:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2} F_i (\theta_i - \theta^*_{A,i})^2 \tag{2.3}$$

This technique will be used to support the multi-label VAG model, combining the two techniques to achieve better performance.

2.3 LAMOL

LAMOL (LAnguage MOdeling for Lifelong Language Learning) (13) introduces a novel approach to Continual Learning and combines language modeling (LM) and question answering (QA) tasks into a single model, and it basically generates pseudo-old samples to prevent catastrophic forgetting.

The QA tasks uses the context and tries to generate a correct answer, LM tasks try to predict the next word or token given the previous words or tokens. It introduces a mechanism to generate pseudo-old samples. This strategy involves creating samples that emulate the data distribution of previously encountered tasks before initiating training on a new task. By employing top-k sampling, the model constructs these pseudo samples, representing past tasks $(T1, T2, ...)$ up to the current task Ti. This preemptive measure ensures that the model retains knowledge from previous tasks while adapting to new ones. The new loss function generated by the LAMOL is:

$$L_{TOTAL} = L_{\text{QA}} + \lambda L_{\text{LM}}. \tag{2.4}$$

L_{QA} is the loss obtained by the question-answering task, while L_{LM} is obtained by the language modeling task.

This framework is typically used for TIL systems but can be adapted for CIL systems.

2.4 PAGeR

PAGER (Prompt Augmented Generative Replay via Supervised Contrastive Learning) (14) addresses the challenge of Continual Learning fine-tuning a model based on different techniques. PAGER follows a process of joint fine-tuning for Lifelong Intent Detection (LID) and Labelled Utterance Generation (LUG) tasks. This involves using a mix of labeled and pseudo-labeled data and generating pseudo-labeled utterances for old intents based on stored prompts. Additionally, knowledge distillation transfers knowledge from the previous model to the current one. The primary goal is to minimize the overall loss by combining all of the previously mentioned techniques:

$$L_{TOTAL} = \lambda_1 L_{\mathrm{ID}} + \lambda_2 L_{\mathrm{KD}} + \lambda_3 L_{\mathrm{R}} + \lambda_4 L_{\mathrm{SCT}}, \tag{2.5}$$

where L_{ID} is the Lifelong Intent Detection Loss, L_{R} is the Labelled Utterance Generation loss, L_{KD} is the Knowledge Distillation loss, and L_{SCT} is the Supervised Contrastive Training loss.

2.5 ACM

The Adaptive Compositional Modules (ACM) (15) are other types of modules used in Continual Learning, based on the Adapter (16), another technique able to add modules for the next tasks. The ACM is based on two different stages: the Decision Stage and the Training Stage. The first stage is used to understand whether a new model should be added or is sufficient to modify an old one. This is made by considering all the outputs of the modules, and, using this information, a new possible module is added. For the Training Stage, pseudo-experience replay is used to avoid CF. The old modules are fine-tuned, and the new models are trained from scratch.

CHAPTER 3

FIRST EXPERIMENTS AND ATTEMPTS

Several unsuccessful or insufficient trials were conducted before arriving at the final techniques explained earlier, which will be described here.

These experiments will not be included in the next chapter (VAG Improvements) because the accuracy was not sufficiently good to define them as an improvement, but it's important to describe them in order to understand the work that has been done and to arouse possible future works for the lecturers.

3.1 New single labels for VAG

The initial experiment involved replacing the original labels of the Clinc150 and Banking77 datasets with the summaries generated from the training sentences. The summaries of varying lengths, similar to the multi-label VAG, were used, but a single label did not provide accurate results. The original labels are effective in describing each sentence using a limited number of vocabulary tokens, making it challenging to improve the VAG with only one instance.

3.2 Key-words instead of labels

An experiment was conducted in this project to use keywords instead of labels for each class. The aim was to ask a GPT to extract these keywords from the training dataset. Different lengths of labels were tried, but the result did not meet expectations. One of the main reasons for the poor accuracy could be related to the sentence transformer, as it may not correctly capture the meaning of a phrase given some words without syntax and context.

3.3 Jaccard similarity for inference

One experiment that aimed to improve the VAG model with a single label involved introducing a new Jaccard similarity measure instead of using cosine similarity for inference. The Jaccard similarity measures the similarity between two sets by dividing the size of their intersection by the size of their union. It ranges from 0 (no overlap) to 1 (identical sets).

It was unnecessary to use a sentence transformer; the sentences were processed directly. However, this change had a negative outcome, since the sentence transformer plays a crucial role for the outcome of the model.

3.4 Labels sharing the LM-head.

In this case, the idea of more labels for each sentence in the dataset is already introduced, and just like the multi-label VAG which will be shown in the next chapter, one or more labels of each class are generated using a GPT, but using a single instance of the model.

The very first idea of the multi-label VAG was to use the same LM-head for each label, such that they share the same neural network architecture. When calculating the multiple losses, they were used to update the weights of one single LM-head. The results were not satisfactory when using the same size of the neural network head with the same number of layers and parameters. This could be because the different losses work on the same weights but in a negative way overlapping the previous updates.

CHAPTER 4

VAG IMPROVEMENTS

Several studies are presented that showcase different techniques and approaches aimed at enhancing the final accuracy of the VAG model. Three main experiments are highlighted, each interconnected as they progressively improve upon the previous one:

1. The initial approach involves integrating a multi-label dataset, given more than one set of labels for each class, with additional instances of the BART model. Each model will handle a different set of labels. This step expands the dataset and introduces more complexity to the learning process. Initially, a single label for each sentence will be added, working with a total of two labels for each sentence and then two instances of the model. Then, three total labels will be used, showing a more stable result than the previous one.

2. A new, more complex, and powerful sentence transformer will be tested, demonstrating that employing a dataset with more articulate sentences as labels can enhance the final accuracy. This improvement occurs because embedding the meaning of the phrase becomes easier.

3. In the end, a combination of two techniques, VAG and EWC, will be demonstrated. These two types of models are applied to a Two-Model dataset to calculate the final loss. This process yields the best accuracy among all the methods reported here.

4.1 Datasets

For all the studies, two distinct datasets were used to conduct comprehensive experiments and evaluate the effectiveness of various techniques. These datasets played a crucial role in our research, primarily focusing on the domain of text classification. In the classification task, each sentence within the datasets is associated with a corresponding class label. The goal of employing these datasets in our model is to accurately assign each sentence to the appropriate category, thereby testing the efficacy of the proposed classification techniques.

Both of these datasets were originally used in the VAG experiments, making it possible to compare the outcomes obtained from different proposed techniques.

The two datasets are: Banking77 (17) and Clinc150 (18).

- The Banking77 dataset comprises a collection of 7 distinct tasks, each task further divided into 10 classes.
- The Clinc150 dataset includes a broader spectrum of 15 tasks, again with 10 classes each.

4.1.1 New labels for the dataset

Both Clinc150 and Banking77 datasets have been suitably modified by adding one or more labels to ensure they have at least two. The newly added labels will include a brief class summary with a specific maximum word limit. One of the main objectives of this project is to investigate whether the label's length can affect the model's accuracy and, if so, how much. All the experiments will be conducted using labels of four different lengths: 3, 5, 7, and 10 words.

A generative AI language model based on GPT (10) (19) was used to generate new labels for each class. The model was trained using some sentences from the training dataset and was asked to summarize them with a given maximum length.

The quality of the summary is crucial for the model's final outcome. Various types of summaries have been attempted, such as only keyword usage, question generation, or generating sentences similar to the original training dataset. The best solution is to make the summary as simple as possible, avoiding synonyms or different words (generative AI models usually rely on synonyms) in the training set since it will increase the number of words used from the vocabulary.

4.2 Multi-Label VAG

The first experiment made to enhance the performance of the VAG model in Continual Learning is to use the newly created multi-label datasets for a new complex VAG model with more than one BART instance. This can be accomplished by fine-tuning more pre-trained models using different losses and labels for each class. It will be shown how combining those instances of the VAG model can increase the final accuracy. Figure 8 shows the idea of a Multi-Label VAG with two instances. The figure and the meaning of the Multi-Label VAG will be better described in the next subsection.

In the training process, the multi-label VAG behaves in a similar manner to the original VAG. However, each instance of the model is fine-tuned, which results in the calculation of one cross-entropy loss for every instance. Finally, these losses are used together to update the weights of the model.

During inference, given N the number of the used models and $y_{gen,n}$ the generated labels by the model n, the Equation 2.1 will be obtained by:

$$y_{pred} = \underset{y \in P}{\text{argmax}} \sum_{n \in N} (\cos(\text{embed}(y), \text{embed}(y_{gen,n}))). \tag{4.1}$$

To predict the label for a given sentence, the cosine similarity is calculated between each entry of the label pool of all previously seen labels and the label ($y_{gen,n}$) generated by the model using the input sequence, similar to the original VAG. However, the prediction is now given by the sum of the results of each instance, and the best label is found in the end.

The original VAG code (20) has been modified to allow the inclusion of new semantic labels. For simplicity, let's consider the VAG model with only two labels. Some methods and variables have been duplicated, ensuring that different variables are used for each label. The process can be extended for more than two labels; in fact, a VAG model with three labels will be shown later. The description of the code is better analyzed in one of the next chapters.

4.2.1 Two-Label VAG

Initially, two labels were assigned for each class, and two instances of the BART model were used, so $N = 2$. Each instance has its own label pool of classes, masked vocabulary, and learning model head for all the tasks. Labels for different training sets of various sizes are used to demonstrate how the accuracy is affected by the length of the class. The VAG model masks unused vocabulary tokens; therefore, the accuracy during training is significantly influenced by the size of the labels. However, the quality of each summary can affect this result.

Generally, shorter summaries lead to better results. But if the summary is too short, it may decrease the quality of the labels and accuracy due to the lack of context.

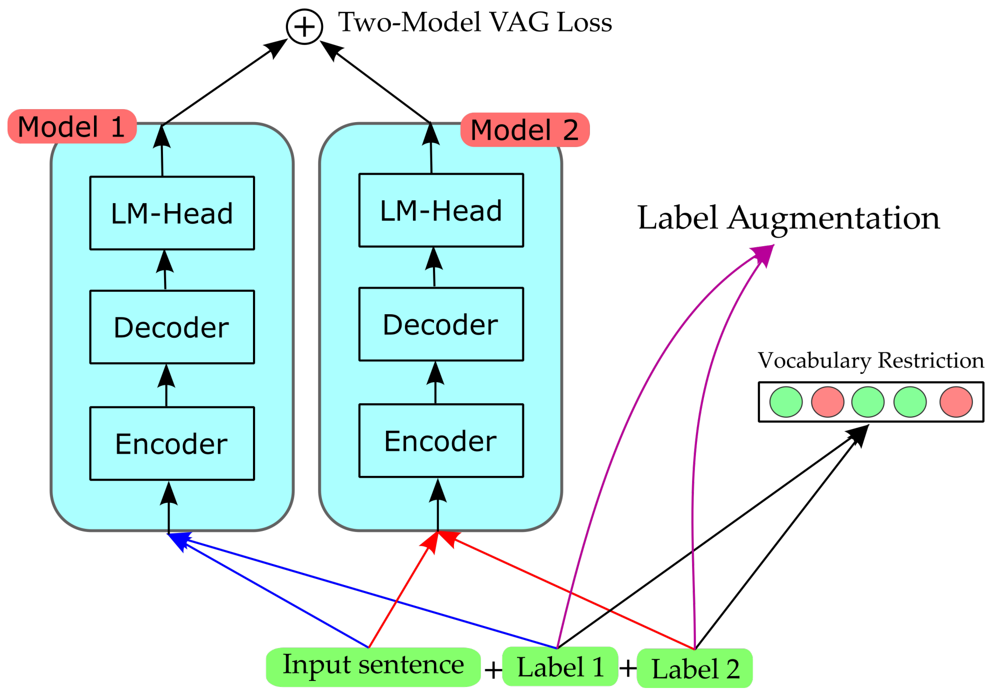


Figure 8: General outlook of the Two-Model VAG during the training part.

Figure 8 illustrates how the Two-Model VAG works. The first model uses the input sentence and the first label, while the second model uses the input sentence and the second label. Both labels are also exploited for the original VAG techniques: Label Augmentation for increasing the dataset and repeating the classes of the previous tasks and Vocabulary Restriction for increasing the next-token probability, just like explained before.

TABLE I: TABLES SHOWING THE ACCURACY USING THE TWO-MODELS VAG WITH TWO DIFFERENT DATASETS

Banking 77

	10 Words	**7 Words**	**5 Words**	**3 Words**
Original Label Not Included	0.361	0.317	0.437	0.369
Original Label Included	0.492	0.619	0.6025	0.408

Clinc150

	10 Words	**7 Words**	**5 Words**	**3 Words**
Original Label Not Included	0.51	0.593	0.647	0.606
Original Label Included	0.693	0.654	0.67	0.648

Table I displays the variation in accuracy based on label length and whether one of the two labels is the original one.

The original VAG model's accuracy score is 0.55 for the BANKING77 dataset and 0.66 for the CLINC150. If the original label is excluded from the dataset, the accuracy decreases, while including it leads to a slight improvement. The second label is considered as a reference to support the original label in choosing the right class and enhance the model's accuracy.

The following experiment may lead us to some important considerations: when creating summaries using the BART model, it's important to avoid making them too short or too long. If they're too short, the model might generate overly simplistic sentences, which can add more errors to the cosine similarity. On the other hand, if the summaries are too long, they can add too many tokens to the masked vocabulary, which can also cause issues in the model's performance.

4.2.2 Three-Label VAG

To assess whether increasing the number of models improves accuracy, a three-label VAG model is introduced. This will help determine how adding more models impacts overall accuracy. Table II shows how the accuracy varies using different label lengths, just like the model before.

TABLE II: TABLE SHOWING THE ACCURACY OF THE THREE-MODEL VAG, JUST LIKE THE PREVIOUS CASE. THE ORIGINAL ACCURACY OF THE VAG MODEL IS SHOWN TO DEMONSTRATE ITS IMPROVEMENTS.

Banking77

	Original VAG	**10 Words**	**7 Words**	**5 Words**	**3 Words**
Original Label Not Included	——	0.484	0.466	0.606	0.562
Original Label Included	0.549	0.578	0.572	0.575	0.575

Clinc150

	Original VAG	**10 Words**	**7 Words**	**5 Words**	**3 Words**
Original Label Not Included	——	0.524	0.58	0.627	0.625
Original Label Included	0.657	0.665	0.666	0.684	0.666

It is worth noting that the accuracy may be lower, in some cases, than the Two-Label VAG; however, when the original label is included, it leads to a more consistent accuracy across different label lengths.

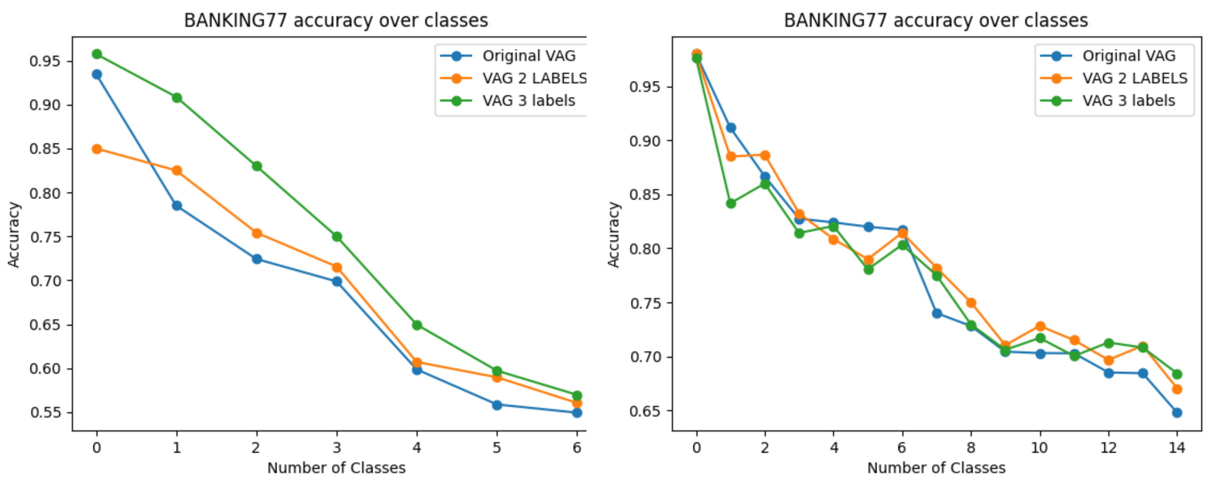


Figure 9: Graphs showing how the accuracy varies increasing the number of tasks for both BANKING77 and CLINC150 datasets, comparing the original VAG accuracy with respect to the Two-Label VAG and the Three-Label VAG.

The experiments were conducted using two or three labels per class. However, increasing the number of labels could reveal whether accuracy improves due to higher precision or decreases due to possible mispredictions. This experiment can be a possible future work, trying to understand if and which is a good number of labels for each class.

Figure 9 shows the progress of the accuracy w.r.t. the classes. The Three-Model VAG has better final accuracy, as shown by Table II too. The Two-Model and Three-Model VAG curves are created by averaging the results of the sentences with 3, 5, 7, and 10 words. Just like the original VAG has shown more room for improvement than the other CL methods, the Multi-Label VAG could even offer a wider opportunity for future research due to less forgetting.

4.3 A new Sentence Transformer

A Sentence Transformer is a neural network architecture that transforms sentences into vector representations, capturing their semantic meaning. As explained before, a sentence transformer is used by the VAG model during the inference part. The label pool of all the previously seen labels and all the outputs of the model are mapped using the transformer to compare with the cosine similarity.

The sentence transformer used by the original VAG model during inference is *paraphrase-MiniLM-L6-v2*, which converts each phrase into a 768-dimensional dense vector. Since in the multi-label VAG, more complex sentences are used, a better embedding can be able to capture more context with more detailed data, giving a better accuracy.

A new embedding, $all-mpnet-base-v2$ (link: `https://huggingface.co/sentence-transformers/all-MiniLM-L6-v2`), has been introduced to demonstrate how the accuracy can be improved with the use of a multi-label VAG. For the sake of simplicity, just the Two-Label VAG has been employed, which is sufficient to demonstrate the improvement in accuracy on this dataset.

TABLE III: RESULTS OBTAINED USING THE NEW SENTENCE TRANSFORMER "ALL-MPNET-BASE".

	Original VAG	**VAG + new transformer**	**10 Words**	**7 Words**	**5 Words**	**3 Words**
Banking77	0.552	0.534	0.553	0.615	0.6	0.561
Clinc150	0.657	0.661	0.706	0.67	0.686	0.664

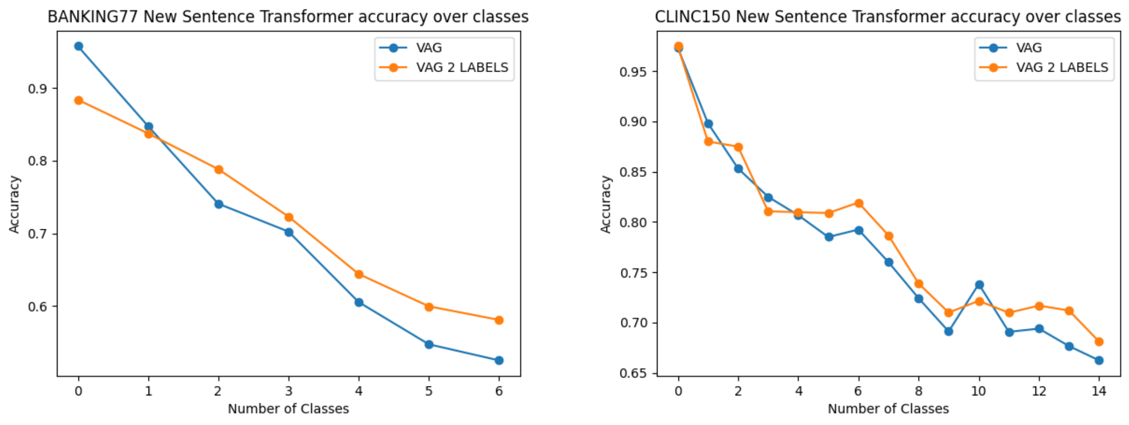


Figure 10: Graphs showing how the accuracy varies, increasing the number of tasks for both BANKING77 and CLINC150 datasets, comparing the original VAG accuracy with respect to the Two-Model VAG with a new sentence transformer.

Table III demonstrates that the new sentence transformer doesn't improve the accuracy of the original VAG model, instead, it decreases. However, the new sentence transformer improves accuracy when paired with the Two-labels model (then with more complex phrases). This is because the new sentences used as labels provide richer context compared to the simple keywords used in the original model, and the new transformer provides richer quality than the first one (source: `https://www.sbert.net/docs/pretrained_models.html`). Additionally, the next chapter will show how the VAG+EWC model with the new sentence transformer achieves the highest accuracy among the models compared.

Similar to the comparison shown in Figure 9, Figure 10 displays a comparison between the VAG and the Two-Model VAG models that utilize the new sentence transformer. It focuses on the variation of the accuracy across multiple tasks, which again exhibits the new model's superiority.

4.4 VAG-EWC: two-model combination

An enhancement that can be implemented to improve the accuracy of the VAG model involves integrating it with EWC. This combination aims to preserve the performance of older tasks while still allowing for improvements on new tasks, thus striking a balance between retaining learned information and adapting to new data. In this scenario, the first instance of the model uses the VAG-loss and label pseudo-replay; the second instance of the model exploits the EWC technique. This is achieved through the use of a fisher buffer, which helps maintain some good weight values.

For the sake of simplicity, a two-model VAG is shown, characterized by a dataset with the original class label and a general summary sentence, to demonstrate the improvement over the original VAG.

Figure 11 shows how the new VAG-EWC model works in broad outlines. Just like the Two-Model VAG explained before (Figure 8), the input sentence and the two labels are respectively passed to the two models during training, exploiting the Vocabulary Restriction and the Label Augmentation. The second instance of the model uses the fisher matrix of the EWC technique to maintain the best weights for the tasks unchanged. The new loss is then calculated, updating the necessary weights.

The Table IV shows how the final accuracy in the two datasets is better w.r.t. the original VAG accuracy.

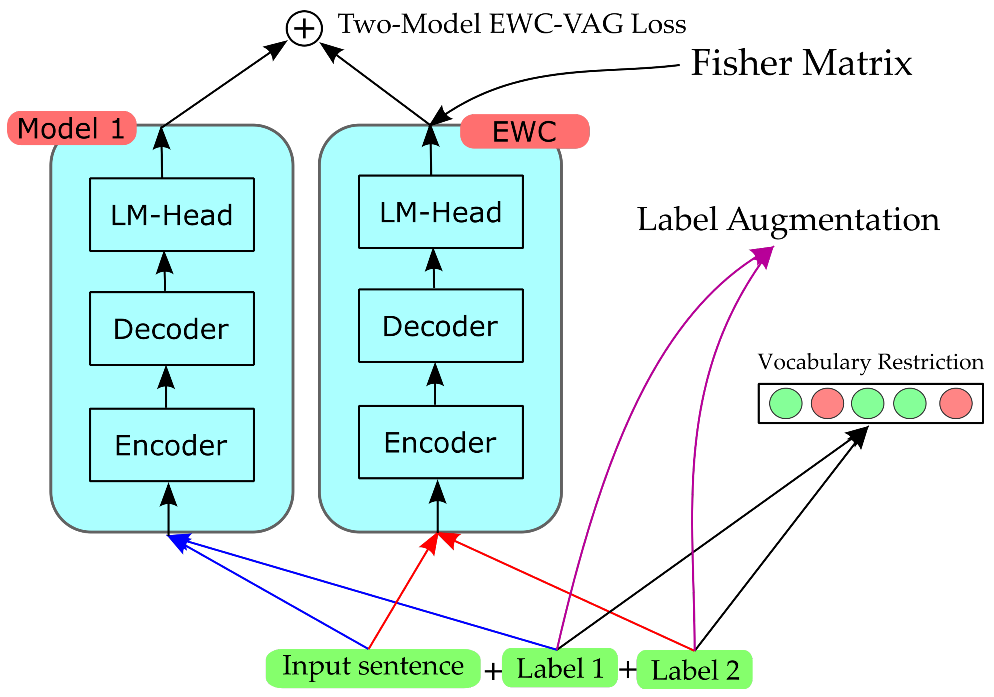


Figure 11: General outlook of the Two-Model VAG during the training part.

TABLE IV: RESULTS OBTAINED USING THE VAG-EWC MODEL, THAT COMBINES THE TWO TECHNIQUES.

	Original VAG	**10 Words**	**7 Words**	**5 Words**	**3 Words**
Banking77	0.552	0.656	0.619	0.647	0.616
Clinc150	0.657	0.71	0.698	0.7	0.677

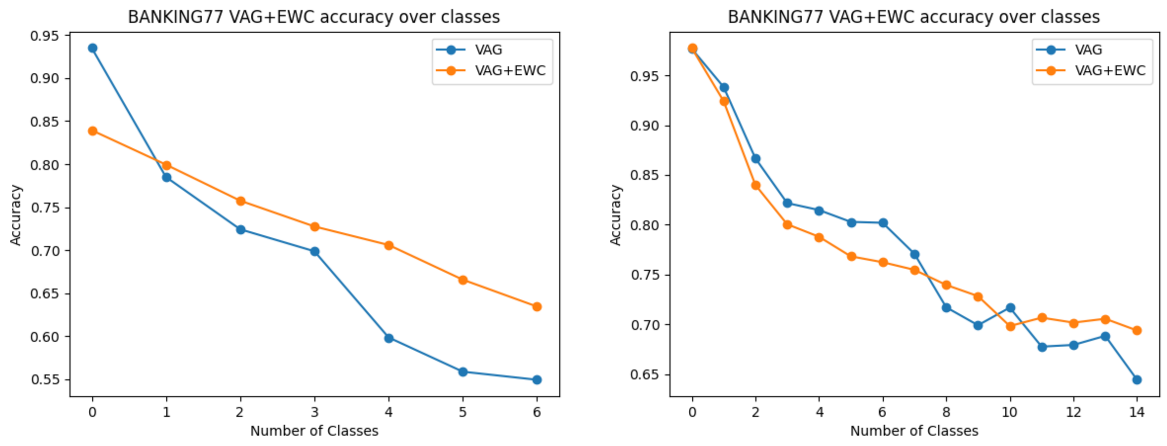


Figure 12: Graphs showing how the accuracy varies, increasing the number of tasks for both BANKING77 and CLINC150 datasets, comparing the original VAG accuracy with respect to the Two-Label VAG and the Three-Label VAG.

For completeness, Figure 12 shows the performance of the VAG-EWC w.r.t. the original VAG. The improvement is much more evident here, especially in the BANKING77 dataset.

CHAPTER 5

IMPLEMENTATIONS

This chapter will focus on how the previously cited techniques are implemented, analyzing the modified parts of the code.

As was announced before, the skeleton of the model is the code of the original VAG model (20). It will be properly modified to accommodate the new techniques presented before.

The original VAG project comprises multiple files and lines of code, making it difficult to upload added or modified code here. Providing a theoretical description of the changes and implementations would be more appropriate. Many functions and parts of code are from the original VAG, and they just need an adaptation for handling more than one label.

5.1 Reading the new dataset

First, the new label set should be integrated to be used during both the training and inference part. The original dataset is composed of three parts: text (the input sentence), label (the number corresponding to the class), and semantic label (the real phrase, since VAG uses it). The only thing that should be added is one or more semantic labels, depending on the number of labels that should be added, like this example of Three-Model VAG:

```
data = {'train': {'text': [], 'semantic_labels': [],'semantic_labels1': [], '
    semantic_labels2': [], 'labels': []},
'test': {'text': [], 'semantic_labels': [],'semantic_labels1': [], '
    semantic_labels2': [], 'labels': []},
'dev': {'text': [], 'semantic_labels': [], 'semantic_labels1': [], '
    semantic_labels2': [],'labels': []}}
```

Depending on how the new labels are added, the dataset should be filled with the new data (more information on the new dataset can be found in the Appendices). Once the dataset and corresponding label sets are correctly filled, new data must be added due to the label replay (augmentation) technique. Given a class, augmentation will be applied for each label.

The original VAG uses a preprocess function for handling the tokenizer, padding, and mapping from labels to indices for the targets. This function can be adapted to accommodate more labels. For initializing the test/training setup and vocabulary restriction, the code is multiplied based on the number of labels for each class. The vocabulary is then masked based on all the labels, including the new ones.

5.2 Training Part

The training part of the original VAG, which can be considered as the fine-tuning of a pre-trained BART model, is divided into different nested files. The innermost class is the heart of the model since it is the modified BART in which the LM-Head is contained. This class will be duplicated for the Two-Models VAG, replicated for the Three-Model VAG, and so on. It is also the least modified class. Here, a sanity check of the restricted vocabulary is made, and, as it was said before, a vocabulary for each set of labels is given. Once the input sentence and label are provided, the ”LM logits” are calculated. These are the model’s raw output values before applying the softmax function. Finally, the loss is calculated based on the masked vocabulary.

The instance of the previous class is called and created by another outer class that can be considered as a sort of wrapper, combining the possible multiple instances of the Multi-Vag model and containing the information related to the masked vocabularies, one for each set of labels, and some functionalities related to the label replay of the VAG and the fisher matrix for the EWC. This class takes all the outputs of the Multi-Label VAG given by the personalized BART models and packs them for the calling function in order to use the loss for updating the weights. For the Multi-Label VAG, some parts of the code should be multiplied to handle each vocabulary and augmented labels, while the instances of the class are easily extended like this:

```
self.model = MyBartForConditionalGeneration.from_pretrained(args.
    model_name_or_path, args=args)
self.model2 = MyBartForConditionalGeneration.from_pretrained(args.
    model_name_or_path, args=args)
self.model3 = MyBartForConditionalGeneration.from_pretrained(args.
    model_name_or_path, args=args)
```

In this case, for example, a Three-Model VAG is used, handling independently and separately each instance.

The last part of the training structure is a function, calling the previous class and passing the input (sentence and multi-labels) to calculate the output, like this example for the Three-Label VAG:

```
outputs, outputs1, outputs2 = model(**inputs)
```

Those outputs are then used for the loss (both original loss and the label replay loss, calculated by the augmented labels), applying the backward propagation:

```
ori_loss = ori_loss = outputs.loss + outputs2.loss + outputs3.loss
label_replay_loss = outputs.label_replay_loss + outputs2.label_replay_loss +
    outputs3.label_replay_loss
loss = loss + self.args.lamb * label_replay_loss
```

For the VAG-EWC model, the approach is the same as the Multi-Label VAG, but one of the two instances of the model is modified to handle the fisher matrix.

5.3 Inference part

The inference stage is the final step, and the major modifications related to the Multi-Label VAG (such as adding new instances, modeling, and sharing functions) have already been covered since they are the same as the training stage. As mentioned earlier, during the original VAG inference, the cosine similarity is used to identify the best label from all of the previously seen labels while comparing it with a label generated by the input test sentence.

In order to generate multiple labels, the code must be adjusted such that each input sentence can produce one label per model. For example, given a Three-Label Model, a prediction of the new label is:

```
# generating outputs
outputs = self.model.generate(input_ids)
outputs1 = self.model1.generate(input_ids)
outputs2 = self.model2.generate(input_ids)
# decoding outputs
decoded_labels = self.tokenizer.batch_decode(outputs, skip_special_tokens=True)
decoded_labels1 = self.tokenizer.batch_decode(outputs1, skip_special_tokens=
    True)
decoded_labels2 = self.tokenizer.batch_decode(outputs2, skip_special_tokens=
    True)
# using embedder
embeds_for_retrieving = torch.tensor(self.embedder.encode(decoded_labels))
embeds_for_retrieving1 = torch.tensor(self.embedder.encode(decoded_labels1))
embeds_for_retrieving2 = torch.tensor(self.embedder.encode(decoded_labels2))
# calculate cosine similarity
cosine_sim = sim_matrix(embeds_for_retrieving, self.label_pools)
cosine_sim1 = sim_matrix(embeds_for_retrieving1, self.label_pools1)
cosine_sim2 = sim_matrix(embeds_for_retrieving2, self.label_pools2)
# summing cosine similarities
cosine_sim = cosine_sim + cosine_sim1 + cosine_sim2
# predict new label
predictions = cosine_sim.argmax(dim=1)
pred_label = self.label2idx.to(predictions.device)[predictions]
```

First of all, the outputs are generated given the input sentence and the different models. Then, it is decoded, obtaining the real phrase instead of numbers (tokens). A sentence transformer is used to encode the newly obtained labels, and finally, the cosine similarity is calculated using a label pool formed by all the classes seen before. The final array is calculated by summing all the similarities, and the best label is predicted.

CHAPTER 6

CONCLUSION

This thesis's main objective is to contribute to the progress of Continual Learning. Our focus was on the VAG algorithm, which employs a combination of techniques from machine learning and NLP and leverages new methods developed specifically for the model.

First, we modified the two datasets used by the original VAG for classification by adding more labels for each class. These new labels summarize the different training sets, enhancing the dataset.

We experimented with varying sizes for the label summaries. The main objective of this experiment was to demonstrate how the accuracy and effectiveness of the model are affected by the number of words used for each label. This can be due to either the inadequate class description or an excessive number of words used in the vocabulary.

The modified VAG model can now simultaneously handle multiple labels for each class, increasing the number of instances and fine-tuning them. We tested a Two-Labels VAG and a Three-Label VAG, both of which showed improved accuracy.

We also improved the Multi-Label VAG by changing the sentence transformer used. Finally, we combined the VAG model and the EWC model, resulting in the best average result among all the experiments.

The thesis presented multiple ideas, each suggesting avenues for future works. A first possible future work is to modify the BART pre-trained model, especially the LM-head size, trying to increase it in order to use a single instance of the model while working with a bigger neural network. Due to several constraints that required more time, similar work was attempted without altering the size of the LM-Head as mentioned in Chapter 3's First Experiment And Attempts.

Different sentence transformers could be tested to improve the accuracy of the summaries' meaning capture, and other methods than the cosine-similarity can be tried, like Euclidean or Minkowski distance.

Finally, one of the major tasks for the future is to automate Multi-Label VAG. This can be achieved by using an already fine-tuned generative AI model that can create labels from the training set and directly improve the dataset.

This thesis sheds light on the intricate dynamics of Continuous Learning and Lifelong Learning and could open the way for different research opportunities in this fascinating field. It has significantly heightened our interest and curiosity. We hope that the insights and findings presented herein will serve as a valuable resource and a source of inspiration for those looking to contribute to and possibly achieve breakthroughs in the realm of Continuous Learning.

APPENDIX

EXAMPLES OF THE NEW DATASETS

A short part of the two datasets, properly modified for the Multi-Label VAG, Banking77 and Clinc150, will be shown here. The format of the two dataset is preserved, only the labels are modified.

A.1 Banking77 dataset format

Input sentence	Labels
I ordered a card but it has not arrived. Help please!	card_arrival – Card delivery status: still pending – Card delivery status tracking assistance
I think someone else is using my card. There are transactions I didn't make.	compromised_card – Unauthorized transactions; suspect card fraud – Suspected unauthorized card usage reported
Why did I get charged a fee when I tried to obtain cash?	cash_withdrawal_charge – Fee charged for cash withdrawal – Surprised by ATM withdrawal fees

The Banking77 dataset is available in a CSV file format, separated into two files (train and text) and consisting of two columns - the input sentence and the original label.

For the Multi-Label VAG, the same format and number of rows and columns are used, but the second column is extended to have more than one label. The labels are separated by a particular set of characters, which in this case is ”−−”, as shown in the table. The separated labels are then used in the code as explained in the previous chapters.

A.2 Clinc150 dataset format

Data file:

```
{"val":[
    [   "in spanish, meet me tomorrow is said how",
        "translate -- Seek language translations. -- Translation assistance required."  ],
            ...
        ],
 "train":[
    [  "ai, show me online options to order more checks for my usbank account",
            "order_checks -- Ordering additional checks -- Reordering checkbooks"   ],
        ],
 "test":[
    [   "if i lose my job will my credit score go down",
            "improve_credit_score -- Boost credit score strategies -- Improve credit rating methods"    ],
            ...
        ]
}
```

Labels file:

```
{
...
"improve_credit_score -- Boost credit score strategies -- Improve credit rating
    methods": 1,
...
"translate -- Seek language translations. -- Translation assistance required.":
    35,
...
"order_checks -- Ordering additional checks -- Reordering checkbooks": 105,
...
}
```

The Clinc150 dataset comes in two JSON files: one file includes all the data, and the other file contains the labels. The dataset is divided into three sets: validation, training, and test. Each set has various sentences assigned to different classes. Every array inside the sets is two-dimensional, where the first dimension represents the input sentence, and the second one is the label. Like Banking77, each array's second dimension has more labels for each class, separated by a special character set. This extension is done for the labels file too.

CITED LITERATURE

1. Torr, M.: Deep learning – what is it and why does it matter?, 2017. `https://www.marktorr.com/deep-learning/` [Online; accessed May 30, 2024].

2. Opitz, D. and Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research, 11:169–198, August 1999.

3. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I.: Attention is all you need, 2023.

4. Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L.: Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

5. Shao, Y., Guo, Y., Zhao, D., and Liu, B.: Class-incremental learning based on label generation, 2023.

6. Ke, Z. and Liu, B.: Continual learning of natural language processing tasks: A survey, 2023.

7. McCloskey, M. and Cohen, N. J.: Catastrophic interference in connectionist networks: The sequential learning problem. volume 24 of Psychology of Learning and Motivation, pages 109–165. Academic Press, 1989.

8. Kim, G., Xiao, C., Konishi, T., Ke, Z., and Liu, B.: A theoretical study on solving continual learning, 2022.

9. Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

10. Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D.: Language models are few-shot learners, 2020.

CITED LITERATURE (continued)

11. Reimers, N. and Gurevych, I.: Sentence-bert: Sentence embeddings using siamese bert-networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, 11 2019.

12. Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., Hassabis, D., Clopath, C., Kumaran, D., and Hadsell, R.: Overcoming catastrophic forgetting in neural networks. Proceedings of the National Academy of Sciences, 114(13):3521–3526, March 2017.

13. Sun, F., Ho, C., and Lee, H.: LAMAL: language modeling is all you need for lifelong language learning. CoRR, abs/1909.03329, 2019.

14. Varshney, V., Patidar, M., Kumar, R., Vig, L., and Shroff, G.: Prompt augmented generative replay via supervised contrastive learning for lifelong intent detection. In Findings of the Association for Computational Linguistics: NAACL 2022, eds. M. Carpuat, M.-C. de Marneffe, and I. V. Meza Ruiz, pages 1113–1127, Seattle, United States, July 2022. Association for Computational Linguistics.

15. Zhang, Y., Wang, X., and Yang, D.: Continual sequence generation with adaptive compositional modules. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), eds. S. Muresan, P. Nakov, and A. Villavicencio, pages 3653–3667, Dublin, Ireland, May 2022. Association for Computational Linguistics.

16. Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S.: Parameter-efficient transfer learning for NLP. In Proceedings of the 36th International Conference on Machine Learning, eds. K. Chaudhuri and R. Salakhutdinov, volume 97 of Proceedings of Machine Learning Research, pages 2790–2799. PMLR, 09–15 Jun 2019.

17. Casanueva, I., Temčinas, T., Gerz, D., Henderson, M., and Vulić, I.: Efficient intent detection with dual sentence encoders. In Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, eds. T.-H. Wen, A. Celikyilmaz, Z. Yu, A. Papangelis, M. Eric, A. Kumar, I. Casanueva, and R. Shah, pages 38–45, Online, July 2020. Association for Computational Linguistics.

18. Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A., Tang, L., and Mars, J.: An evaluation dataset for intent classification and out-of-scope prediction. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing

CITED LITERATURE (continued)

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), eds. K. Inui, J. Jiang, V. Ng, and X. Wan, pages 1311–1316, Hong Kong, China, November 2019. Association for Computational Linguistics.

19. Yenduri, G., M, R., G, C. S., Y, S., Srivastava, G., Maddikunta, P. K. R., G, D. R., Jhaveri, R. H., B, P., Wang, W., Vasilakos, A. V., and Gadekallu, T. R.: Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions, 2023.

20. Shao, Y., Guo, Y., Zhao, D., and Liu, B.: Class-incremental learning based on label generation. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), eds. A. Rogers, J. Boyd-Graber, and N. Okazaki, pages 1263–1276, Toronto, Canada, July 2023. Association for Computational Linguistics.

21. Goodfellow, I. J., Bengio, Y., and Courville, A.: Deep Learning. Cambridge, MA, USA, MIT Press, 2016. `http://www.deeplearningbook.org`.

VITA

NAME	Giuseppe Gabriele
EDUCATION	
	Master of Science in "Computer Science", University of Illinois at Chicago, May 2024, USA
	Specialization Degree in "Artificial Intelligence and Data Analytics", July 2024, Polytechnic of Turin, Italy
	Bachelor's Degree in Computer Engineering, July 2022, Polytechnic of Turin, Italy
LANGUAGE SKILLS	
Italian	Native speaker
English	Full working proficiency
	2022 - IELTS examination (B2 Level)
	A.Y. 2023/24 One Year of study abroad in Chicago, Illinois
	A.Y. 2022-2023. Lessons and exams in both the Polytechnic of Turin and University of Illinois at Chicago attended exclusively in English
SCHOLARSHIPS	
Fall 2023	Italian scholarship for TOP-UIC students
TECHNICAL SKILLS	
Basic level	General knowledge of MIPS, ARM, Matlab, electronics
Average level	Good knowledge of SQL language, C and C++
Advanced level	Advanced knowledge of Python, PyTorch, Javascript, React-NodeJS
WORK EXPERIENCE AND PROJECTS	
Sep 2022	Erasmus+ Project: "Resilient Youth for a Meaningful Society
Mar 2022 - July 2022	Academic Apprenticeship: JavaScript-React developer for an internet program, still work in progress