Report on Student Research Project

Third Year of French Grande École

ENSTA Paris - Institut Polytechnique de Paris

On the learning capacity of LLM : A concept-oriented study

Authored by :

Mr. Raphaël BERNAS - Student at ENSTA Paris in Applied Mathematics & ENS Paris-Saclay in MVA master

Supervised by :

Dr. Pr. Céline HUDELOT and Dr. Fanny JOURDAN

From April 14^{th}*, to October* 13^{th}*, 2025*

Dr. Pr. Andrea SIMONETTO	ENSTA Paris	- ENSTA Tutor and Examiner
Dr. Pr. Guillaume CHARPIAT	INRIA Saclay	- MVA Tutor and Examiner

Internship at Centrale-Supélec mathematics and informatics laboratory (MICS)

9 rue Joliot Curie, France - 91192 Gif-sur-Yvette - Bâtiment Bouygues

Class of : 2022/2025

Defended on October 20^{th}, 2025.

Non - Confidential

Confidentiality note

This report is non-confidential, thus it can be shared online and kept by ENSTA Paris.

Contents

Acknowledgments

All the works presented here are the results of a simultaneous research lead by Dr. Pr. Céline Hudelot, Dr. Fanny Jourdan and Student Raphaël Bernas.
They would not have been possible without the *Interpreto* library of the DEEL & FOR team from IRT Saint-Exupéry in Toulouse. As well as the *EuroBERT* project of the GEMILA team from MICS at Centrale Supélec.

Personal Thanks

I would like to express my deepest gratitude to the MICS lab for their warm welcome.

I'm specially thankful to Céline and Fanny for this opportunity and for their accompagnement during the entierity of this internship. To Antonin for his help and for his multiple insight. To Adil, Ayoub, Darpan, Gabriel, Hélène, Julien, Marcos, Oussmane, Rachel, Romain, Vincent and all the others member of the MICS whom acompanied me during this internship.

I would like to express my deepest gratitude to my family and my school, ENSTA Paris (with the Institut Polytechnique de Paris), as well as to the direction of the MVA master and ENS Paris-Saclay for their support.

Finally, I would like to once again express my gratitude to the MICS & Centrale Supélec and specially to Fabienne Brosse & Leila Sidali, without whom this internship would not have been possible.

Information

Résumé

L'explicabilité (XAI) est née du besoin de rendre les prédictions des systèmes d'intelligence artificielle plus transparentes et interprétables, en particulier à mesure que les modèles gagnent en complexité. Ce besoin est particulièrement marqué dans le contexte de l'apprentissage profond (Deep Learning - DL), où les décisions des modèles sont souvent opaques. L'interprétabilité mécanistique, un sous-domaine de la XAI, cherche à relever ce défi en analysant la structure interne et le fonctionnement des grands modèles afin d'expliquer leurs sorties. Cependant, la plupart des approches existantes se concentrent sur une analyse a posteriori, effectuée après l'entraînement, et peu d'entre elles ont exploré l'utilisation de méthodes mécanistiques pour suivre le processus d'apprentissage lui-même. Dans ce travail, nous étudions l'évolution des mécanismes internes d'un modèle au cours de l'entraînement, en utilisant le modèle EuroBERT et ses checkpoints intermédiaires. En appliquant des techniques d'interprétabilité mécanistique tout au long de la phase d'apprentissage, nous visons à extraire des observations sur la dynamique d'apprentissage des grands modèles de langage (LLMs), dans le but de contribuer à l'élaboration de cadres d'apprentissage plus structurés et à l'établissement d'une dynamique de l'apprentissage.

Mots-clés - Apprentissage, LLM, Concepte, Explicabilité, Mechanistique, IA

Abstract

Explainable Artificial Intelligence (XAI) has emerged from the need for transparent and interpretable AI predictions, especially as models grow in complexity. This need is particularly present in the context of Deep Learning (DL), where model decisions are often opaque. Mechanistic interpretability, a subfield of XAI, addresses this challenge by seeking to understand the internal structure and functioning of large models to explain their outputs. However, most existing approaches focus on post-hoc analysis, applied after training, and relatively few have explored the use of mechanistic methods to monitor the learning process itself. In this work, we investigate the evolution of model internals during training using the EuroBERT model and its intermediate checkpoints. By applying mechanistic interpretability techniques throughout the training phase, we aim to derive insights into the learning dynamics of Large Language Models (LLMs), ultimately contributing to the development of more structured learning frameworks and learning dynamics.

Keywords - Learning, LLM, Concept, Explainability, Mechanistic, AI

1 Introduction

Deep Neural Networks (DNNs) have demonstrated remarkable performance over the past decades. However, these achievements have come at a significant cost. In contrast to traditional models in the field of Artificial Intelligence (AI), the behavior of DNNs remains largely opaque and poorly understood. To address this challenge, several research directions have emerged, aiming to provide theoretical and practical insights into these models. Notable among these are studies on robustness [Goo15], formal verification [PT10], and eXplainable AI (XAI) [DSB17]. In this work, we adopt the perspective of XAI. The field of XAI is commonly divided into two complementary areas: interpretability and explainability. Our focus will primarily be on methods stemming from interpretability, and in particular, mechanistic interpretability. Nonetheless, we will also explore aspects of explainability where relevant. Let us now define more precisely the terminology used throughout this work. The definitions below are adapted from the survey *"Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI"* [ADRS+19], while also reflecting recent developments in the field-particularly the emergence of mechanistic interpretability.

Explainability. A collection of methods developed to explain a model's decisions using information that is understandable to humans.

Interpretability. A collection of methods aimed at discerning and identifying distinct internal behaviors exhibited by a model.

The key distinction between these two fields lies in their respective goals. *Explainability* assumes that a model's decisions can be made comprehensible at the human level, often by designing interfaces or abstractions that map internal processes to human-understandable concepts. In contrast, *interpretability* focuses on identifying prediction-dependent mechanisms or behaviors within a model, expecting them to be human-interpretable[1].

Historically, this distinction is rooted in the evolution of machine learning. Explainability gained prominence earlier, particularly in the context of classical machine learning (ML), where models were relatively simple, datasets were smaller, and tasks were more constrained-making post hoc explanations feasible. However, with the rise of DNNs and the increasing complexity of model architectures, the limitations of explainability became more pronounced. As a result, interpretability, which makes weaker assumptions about human alignment, has become an increasingly central focus of research.

In this work, we adopt the definitions provided above. Nonetheless, it is important to acknowledge that there is no

[1] It is not always the case; some method admits that interpretation is tough.

clear consensus in the literature on how to formally delineate these fields. That said, this lack of precision does not hinder our goals, as our focus lies more in the methodology than in rigid terminological boundaries.

Mechanistic Interpretability (MI). Within this context, *Mechanistic Interpretability* has emerged as a subfield focused on understanding large-scale models by studying their internal mechanisms [SW24]. The introduction of transformer-based architectures significantly increased the demand for techniques capable of inspecting model weights and their transformations [VSP+23]. MI originally developed from attempts to reverse-engineer the computations of transformers. While early efforts often targeted computer vision, the field has increasingly shifted toward natural language processing (NLP).

The widespread adoption of transformers in large language models (LLMs), coupled with the attention these models have received, led MI to gradually converge with the broader area of Natural Language Processing Interpretability (NLPI) [SW24]. Today, drawing a strict boundary between MI and other forms of interpretability is increasingly difficult. For the purposes of this work, we define MI as a subset of interpretability methods specifically concerned with analyzing the internal structure and computations of large models (LMs), with the aim of uncovering and characterizing their learned behaviors.

MI encompasses a variety of interpretability techniques, among which concept-based methods are particularly prominent. These approaches attempt to extract meaningful features-commonly referred to as *concepts*-from a model's internal activations, in order to classify or describe its behavior. Importantly, the goal is not necessarily to recover human-interpretable concepts, but rather to identify latent features based on similarity in internal representations or activations.

It is worth noting that, due to these objectives, MI occasionally overlaps with explainability. This overlap contributes to the ongoing difficulty of drawing clear conceptual boundaries between the two fields.

As the field of XAI has matured, a wide array of tools has been developed to study AI systems in depth. However, these tools have been almost exclusively applied in a post hoc manner. That is, most XAI methods are used to analyze models after the training has been completed. This limitation is deeply tied to the historical origins of the field, which was founded with the goal of interpreting pre-trained models rather than models in the process of learning. Moreover, the increasing competitiveness of the deep learning community has led to reduced access to model internals, including training checkpoints, which has further hindered research into learning-phase interpretability. It is therefore regrettable that, despite the proliferation of tools designed to monitor LMs, few have been applied to understanding their learning dynamics during training. The central objective of the present work is to investigate the underlying processes that govern model learning. More specifically, we aim to provide insights into learning dynamics that remain poorly understood, such as the generalization capabilities of DNNs and the recently observed grokking phenomenon[2] [PBE+22].

2 Background

2.1 On LLMs Capacity

To properly introduce the subject of LLMs (Large Language Models), we briefly present the field of Deep Learning. Deep Learning is a subfield of Machine Learning, which is itself a domain focused on leveraging computational resources to train *models* capable of accomplishing specific tasks. In ML, a model is formally defined by the combination of:

[2]Grokking refers to a phenomenon observed in deep neural networks where a model initially appears to overfit-achieving high accuracy on training data while performing poorly on unseen data-but later, often suddenly, begins to generalize well to unseen data. This behavior, which emerges after prolonged training, challenges traditional machine learning theory and remains an open question in research.

- A set of parameters $\theta \in \Theta$,
- A mathematical function $f : \Theta \times \mathcal{X} \to \mathcal{Y}$,

where Θ denotes the parameter space, $\mathcal{X}$ the input space, and $\mathcal{Y}$ the output space. When a model is *used*, this corresponds to the application of the function $x \mapsto f(\theta, x)$.

Naturally, for a given task, both the choice of function f (i.e., the model architecture) and the parameters θ are critical. However, manually specifying model parameters is infeasible, especially when dealing with large-scale datasets $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$. This limitation is addressed by the field of optimization, which frames learning as the search for parameters that minimize a specific objective.

To this end, a *cost function* $\mathcal{C} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$ is defined to quantify the model's performance on a given task. The formal ML problem can thus be written as:

$$\min_{\theta \in \Theta} \mathbb{E}_{(x,y)\sim\mathcal{D}} \left[\mathcal{C}(f(\theta; x), y)\right] \tag{ML}$$

Solving this problem typically involves iterative procedures known as *learning algorithms*. The primary distinction between classical machine learning and deep learning lies in the scale and complexity of the problem. In particular, deep learning involves significantly more expressive model classes.[3]

DNNs (Deep Neural Networks) are typically built as recursive compositions of layers, commonly referred to as *neurons*. Each neuron possesses its own set of parameters, and the aggregation of all neurons' parameters defines the full parameter set of the model. A neuron is traditionally defined as follows:

Definition:

Definition 1. ***Neuron*** – *A neuron is a function $\nu : \mathbb{R}^m \to \mathbb{R}^l$ defined by*

$$\nu(x) = \sigma(Wx + b),$$

where $x \in \mathbb{R}^m$ is the input, $W \in \mathbb{R}^{l\times m}$ is the weight matrix, $b \in \mathbb{R}^l$ is the bias vector, and $\sigma : \mathbb{R}^l \to \mathbb{R}^l$ is a non-linear activation function.

This definition leads us to a classical neural network structure:

[3] The term *complexity* here refers to a well-defined mathematical notion related to the expressiveness of a model class-that is, how rich the class of functions f is. A highly expressive model class can approximate a wide variety of data distributions, provided the right parameters are chosen. For example, if you attempt to separate data using a linear classifier (e.g., a hyperplane), it will struggle with certain patterns-such as separating the center point of a cross-shaped five-point dataset from the rest. In such cases, the linear model has low expressivity.

Definition:

Definition 2. ***Feedforward Neural Network (FCNN)*** – *A fully-connected (or feedforward) neural network of depth L is a function*

$$f_\theta : \mathbb{R}^{n_0} \to \mathbb{R}^{n_L}$$

defined recursively as a composition of layers:

$$f_\theta(x) = h_L \circ h_{L-1} \circ \cdots \circ h_1(x),$$

where each layer $h_\ell : \mathbb{R}^{n_{\ell-1}} \to \mathbb{R}^{n_\ell}$ corresponds to a neuron. The parameters are $\theta = \{(W_\ell, b_\ell)\}_{\ell=1}^{L}$.

A common instantiation of the FCNN is the Multi-Layer Perceptron:

Definition:

Definition 3. ***Multi-Layer Perceptron (MLP)*** – *An MLP is a feedforward neural network in which each layer applies the same non-linear activation function σ (e.g., ReLU or* tanh*) pointwise. Given layer sizes $(n_0, n_1, \ldots, n_L)$, the function defined by the MLP is:*

$$\mathcal{N}_\theta(x) = \sigma(W_L\, \sigma(\cdots \sigma(W_2\, \sigma(W_1 x + b_1) + b_2) \cdots) + b_L),$$

where the parameters $\theta = \{(W_\ell, b_\ell)\}_{\ell=1}^{L}$ consist of weight matrices $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ and bias vectors $b_\ell \in \mathbb{R}^{n_\ell}$.

Remark:

Remark 1. *Under mild conditions on the activation function σ and the hidden layer widths, MLPs are universal approximators-capable of approximating any continuous function on compact subsets of $\mathbb{R}^n$. A foundational first result in this direction (using the sigmoid activation function) was proved by George Cybenko in 1989 [BDVJ03].*

The FCNN class remained a dominant architecture for years.[4] However, the emergence of attention mechanisms has since reduced their prominence. The most notable breakthrough in this domain came with the introduction of the Transformer architecture [VSP+23]:

Definition:

Definition 4. ***Multi-Head Attention*** – *Given an input $H \in \mathbb{R}^{n \times d}$, multi-head attention computes:*

$$MHAtt(H) = Concat(A_1, \ldots, A_h)W^O,$$

where each head A_i is defined as:

$$A_i = softmax\left(\frac{Q_i K_i^\top}{\sqrt{d_h}}\right) V_i,$$

with $Q_i = HW_i^Q$, $K_i = HW_i^K$, $V_i = HW_i^V$, and $W_i^Q, W_i^K, W_i^V \in \mathbb{R}^{d \times d_h}$. The projection matrix $W^O \in \mathbb{R}^{hd_h \times d}$ ensures the output remains in $\mathbb{R}^{n \times d}$.

[4] Basic convolutional networks can also be interpreted as particular cases of FCNNs.

Definition:

Definition 5. ***Transformer*** – *A Transformer is a deep neural network $\mathcal{T}_\theta$ designed to process sequences $(x_1, \ldots, x_n) \in \mathcal{X}^n$. It consists of L layers, each composed of:*

- *A **Multi-Head Self-Attention** mechanism $MHAtt_\ell$,*
- *A **Multi-Layer Perceptron** MLP_ℓ,*
- *Residual connections and Layer Normalization:*

$$h^{(\ell)} = LayerNorm\left(h^{(\ell-1)} + MHAtt_\ell(h^{(\ell-1)})\right),$$

$$h^{(\ell)} = LayerNorm\left(h^{(\ell)} + MLP_\ell(h^{(\ell)})\right),$$

where $h^{(0)}$ is the embedded input (including positional encodings).

Remark:

Remark 2. *The full model defines a function:*

$$\mathcal{T}_\theta : \mathcal{X} \to \mathbb{R}^{n \times d},$$

where θ contains all learnable weights (attention and MLP). Note that Transformers have undergone significant evolution. For example, the MLP part is often replaced by a Feed-Forward Neural Network.

Remark:

Remark 3. *While understanding every detail of Transformer computations is not necessary, it is helpful to remember that a Transformer is essentially a stack of alternating attention mechanisms and MLPs. The attention layers allow the model to focus on relevant parts of the input, while the MLPs perform the main computations. Conceptually, a Transformer can be viewed as a dynamic MLP capable of learning to focus its computational effort.*

Both attention and MLP weights are learned during training-thus they encode task-specific knowledge.

Modern LLMs, which have fueled the recent surge in AI interest, are based on Transformer architectures. For example, OpenAI's GPT series stands for *Generative Pre-trained Transformer* models.

Why are Transformers so central to language modeling? The origin of attention in neural networks can be traced back to earlier work in machine translation tasks [Wikipedia], [BCB16]. These mechanisms were introduced to help models dynamically adjust their focus depending on the context, ensuring more accurate translations. Though earlier RNN-based models implemented forms of attention, the major breakthrough introduced by the transformer architecture was not the concept of attention itself, which was already recognized as essential for processing text and managing long-range dependencies, but rather its scalability. *Attention* mechanisms had long been accepted as a solution to **context overflow**,[5]

[5] Context overflow occurs when a sequence, such as a long sentence or paragraph, is too lengthy for a model to handle effectively. Without an attention mechanism, the model may assign undue importance to frequent but uninformative tokens, such as "the" or "a", which leads to a dilution of meaningful context. Attention mechanisms allow the model to assign greater weight to relevant inputs, mitigating this issue.

a problem where models lose the ability to focus on relevant parts of long input sequences. The transformer architecture made possible the efficient use of attention at scale, thanks to its highly parallelizable computation. This enabled significantly faster training and inference, allowing for the use of GPU-based computation, which further accelerated the learning process.[6]

This scalability marked the beginning of the LLM era, with models designed for language understanding, often featuring billions of parameters.

Despite their empirical success, there is no solid mathematical guarantee supporting the ongoing trend of increasing model size and complexity. In fact, foundational results from statistical learning theory-such as Vapnik's bounds[7]-would suggest the opposite: that overly complex models should underperform due to overfitting.[8] However, these results do not fully capture the efficiency of DNNs. The truth is Vapnik's results focus more on statistical learning rather than all the different aspects of learning (such as optimization-induced learning). It is commonly agreed that neural networks are capable of generalization. The origin of generalization in deep learning remains difficult to pinpoint. It does not appear to stem solely from the specific structure of the model [ZBH+17]. Instead, it is widely believed that optimization algorithms play a central role in this phenomenon [SDBD21]. In particular, the ability of these algorithms to converge toward certain local minima may partially explain why trained models succeed in generalizing to unseen data. [9]

LLMs, in particular, are built from multiple stacked transformer layers, each of which contains a feedforward sublayer. Every such layer produces **activations** in its own activation space. The term *activation* refers to the output of a neural network layer. While its origin can be traced back to neuroscience, the term became widespread in machine learning following the introduction of backpropagation in deep neural networks [RHW86].

Another closely related term is **embedding**. Historically, embeddings became popular with the introduction of Word2Vec, which mapped discrete words into continuous vector spaces [MCCD13]. Over time, the term evolved to refer more generally to the representations produced at different layers of a neural network. In essence, embeddings are activations that serve as learned vector-space representations of the input.[10]

From this perspective, the terms *activation* and *embedding* are often used interchangeably, though embeddings also emphasize the representational role of these vectors. Building on this, the idea of **concept vectors** arises: concept vectors can be understood as forming the basis of embedding spaces [Wikipedia].

Why does this matter for us? Identifying concept vectors is central to concept-based XAI.[11] In our case, however, we are not only interested in extracting concept vectors, but also in tracking their evolution during the training process. Our hypothesis is that phenomena such as generalization or grokking can be better understood by analyzing how these concept vectors evolve.

For example, a recent toy study on a simple diffusion model showed that generalization emerges when the model disentangles previously entangled concepts in its internal representation [POL+24].[12] Inspired by this, our goal is to

[6] GPUs (Graphics Processing Units) excel at parallel computations. If neural network operations, such as matrix multiplications, are carefully designed, the entire model can be efficiently executed in parallel. In contrast, CPUs (Central Processing Units) are typically optimized for sequential computation.

[7] If you want to learn more about Vapnik results on model complexity, please refer to Section 8.5.

[8] Overfitting occurs when a model becomes too tailored to the training data, capturing noise or anomalies instead of general patterns. This leads to poor generalization on unseen data. Larger models are especially prone to overfitting, as they can "memorize" the training data. Statistical theory historically argued in favor of simpler models that, by necessity, must learn compact representations and general patterns rather than idiosyncrasies of the data.

[9] For further discussion and related insights, see this blog post: [Inference].

[10] Concretely, at layer ℓ, the weight matrix transforms the activation from the previous layer $h^{(\ell-1)}$, embedding it into a new representation space defined by that layer.

[11] In practice, much of XAI research focuses on decomposing activation spaces to recover the internal representations of certain input characteristics. This decomposition often yields concept vectors in reduced-dimensional subspaces.

[12] In this experiment, the diffusion model was trained on examples such as "big red dot", "small red dot", and "big blue dot". Generalization

observe generalization phenomena directly through the evolution of concept vectors.

2.2 On Natural Language Processing

Natural Language Processing (NLP) is a subfield of Artificial Intelligence (AI) that aims to enable interaction between computers and humans through natural language. The precise origins of the field are difficult to determine. Some trace its conceptual foundation to Alan Turing's seminal paper *"Computing Machinery and Intelligence"*, which introduced the *Turing Test*-a foundational idea in AI and language understanding.[13]

Early NLP systems were rule-based, relying heavily on handcrafted linguistic rules. These approaches often lacked robustness and failed to generalize, especially in open-domain settings. The subsequent integration of statistical methods improved performance by allowing systems to learn from data, but it was the advent of ML that fundamentally transformed the field.

A pivotal moment came with the introduction of neural networks into NLP by Yoshua Bengio, Réjean Ducharme, et al [BDVJ03], which laid the groundwork for deep learning in language tasks. This transition was significantly accelerated by the introduction of the Transformer architecture [VSP+23], which now forms the foundation for modern LLMs and has driven the recent surge in those models capabilities. The field of NLP encompasses a wide range of tasks, from *text classification* to *sequence labeling*[14], as well as *language modeling* and *text generation*[15] (This is non-exhaustive).

In this work, our primary focus will be on **language modeling**, and in particular on the **Masked Language Modeling (MLM)** task. This is motivated by the fact that we will later introduce *EuroBERT* [BGBA+25], a family of models trained using MLM, which we will rely on throughout this study.

Broadly speaking, language modeling refers to the family of tasks whose goal is to construct a statistical model of a natural language. The objective is to develop a model capable of predicting tokens so as to reconstruct or generate coherent sentences. This is typically achieved by training the model to estimate probability distributions over possible tokens.

The MLM task, specifically, operates by masking a token within a sentence and training the model to correctly predict the missing token. The most common cost function used in this setting is the **cross-entropy loss**, which compares the model's predicted probability distribution against the ground truth distribution (where the original masked token has probability 1, and all others 0).[16]

An illustration of such a task is provided below:

was assessed by checking when the model could also generate "small blue dot", which was never seen during training.

[13] See [Wikipedia], [Medium], and [Stanford Sophomore Project].

[14] Text classification consists of assigning an entire text to one of several predefined classes. For example, one may wish to determine whether a text is a compliment or an insult. Sequence labeling, in contrast, involves assigning a label to each token in a sequence, such as tagging each word with its part of speech.

[15] Although these tasks differ significantly in nature, they all fall within the scope of NLP. Importantly, a model initially trained on a classification task can often be adapted to language modeling through additional training (fine-tuning).

[16] Formally, the cross-entropy loss for token prediction is written as:

$$\mathcal{L}_{CE}(y, \hat{y}) = -\sum_{i=1}^{V} y_i \log \hat{y}_i,$$

where y_i is the true probability distribution over the vocabulary ($y_i = 1$ for the correct token, 0 otherwise), $\hat{y}_i$ is the predicted probability for token i, and V is the size of the vocabulary.

Example of Masked Language Modeling (MLM)

Input sequence:

`EuroBERT is a deep [MASK] network.`

Target (ground truth):

`EuroBERT is a deep neural network.`

Model prediction (distribution at [MASK]):

$$P(\text{word} \mid \text{context}) = \begin{cases} 0.72 & \text{“neural”} \\ 0.12 & \text{“learning”} \\ 0.07 & \text{“language”} \\ 0.04 & \text{“transformer”} \\ \vdots & \end{cases}$$

Loss (cross-entropy at masked position):

$\mathcal{L}_{\text{MLM}} = -\log P(\text{“neural”} \mid \text{context}) = -\log 0.72$

In practice, we will not rely on the prediction head of *EuroBERT*. Our study is instead centered on the **internal behavior** of the model. More precisely, we focus on the intermediate layers and the activations they produce for a given input, since these are likely the components where much of the model's language comprehension emerges.

It is important to note that the prediction head of such models is typically a simple linear projection followed by a softmax[17] layer. By itself, this head is not capable of performing efficient language modeling. Rather, it depends entirely on the rich representations built within the deeper layers of the network. This strongly suggests that the intermediate layers are where the model develops its actual understanding of language.

For this reason, our work will concentrate on analyzing how these internal representations form and evolve during the training process, with the goal of better understanding how the capacity for language comprehension arises within the model.

3 Framework

3.1 On the Learning Framework

Before delving further into XAI, we aim to formally define the framework used to study the learning process. This framework will allow us to precisely define the terminology employed throughout this work. We begin with the following definitions:

[17] The softmax function is commonly used as the final activation in classification tasks. Given a vector $z \in \mathbb{R}^K$, it produces a probability distribution over K classes:

$$\text{softmax}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}, \quad \forall i \in \{1, \ldots, K\}.$$

This ensures that all outputs are non-negative and sum to 1, turning them into probabilities.

Definition:

Definition 6. ***Model Architecture*** *- Let $N \in \mathbb{N}$ denote the depth of the model. Given an input space $\mathcal{X}$, output spaces $(\mathcal{Y}^k)_{k\in\{1,\dots,N\}}$ (with $\mathcal{Y}^0 = \mathcal{X}$), and a parameter space Θ, define $h^k : (y^k, \theta) \in \mathcal{Y}^k \times \Theta \mapsto (h^k_\theta(y^k), y^k) \in \mathcal{Y}^{k+1} \times \mathcal{Y}^k$. A model architecture is then given by:*

$$f_* : (x, \theta) \mapsto \left(h^N_\theta(h^{N-1}_\theta(\cdots), \theta), h^{N-1}_\theta(h^{N-2}_\theta(\cdots), \theta)\right) \in \mathcal{Y}^N \times \mathcal{Y}^{N-1},$$

for $(x, \theta) \in \mathcal{X} \times \Theta$. We will note $\mathcal{NN}$ the set of model architecture.

This recursive definition characterizes the architecture as the concatenation of all the activation layers of a DNN. This definition will come in handy when we will have to perform proof on the activations. Currently, we will only need the following object :

Definition:

Definition 7. ***Decomposable Model*** *- Given an input space $\mathcal{X}$, output spaces $\mathcal{Y}$. We note $f : \mathcal{X} \to \mathcal{Y}$ a decomposable model if and only if there exist g and h such that $f = g \circ h$ and $\mathcal{X} \xrightarrow{h} \mathcal{H} \xrightarrow{g} \mathcal{Y}$ where there exists $k \in \mathbb{N}$ such that $\mathcal{H} \subset \mathbb{R}^k$, the embeddings space.*

Remark:

Remark 4. *Given $f_* \in \mathcal{NN}$ a model architecture and $\theta \in \Theta$, some fixed parameter. Then, $f_*(\cdot, \theta) : x \mapsto f_\theta(x)$ is a decomposable model.*

Definition:

Definition 8. ***Update Rule*** *- An update rule or learning algorithm is a function of the form $\mathcal{A}_L : \mathcal{NN} \times \Theta \times \mathbb{R} \to \Theta$*

Note that in this formulation, the learning objective (e.g., the loss function) is encapsulated within the update rule $\mathcal{A}_L$; thus, we do not refer to the loss explicitly and we won't need to delve deeper into that for our current work.

Definition:

Definition 9. ***Learning Process*** *- Given a model architecture $f_* : \theta \mapsto f_\theta(\cdot)$, a learning algorithm (or update rule) $\mathcal{A}_L$, and an initialization point θ_0, we define a learning process as the tuple $(f_*, \mathcal{A}_L, \theta_0)$ satisfying the iterative update:*

$$\theta_{t+1} = \mathcal{A}_L(f_*, \theta_t, t), \quad \forall t \geq 0.$$

Next, we introduce terminology used to characterize the learning process. All such definitions will be metric-based. The field of interpretability provides a broad range of metrics to monitor a model, and these will serve as tools to understand the model's learning capabilities.

Definition:

Definition 10. ***Learning Dynamics*** *- Given a learning process $\mathscr{L} = (f_*, \mathcal{A}_L, \theta_0)$ with associated parameter trajectory $(\theta_t)_{t\in\mathbb{N}}$ and a metric m, we define the learning dynamics $\mathscr{E}$ via the recurrence:*

$$m(f_{\theta_{t+1}}) = \mathscr{E}(m(f_{\theta_t}), R_t),$$

where R_t is a random variable. If a continuous-time extension of the process is considered[a], and assuming sufficient differentiability, the dynamics may alternatively be defined as:

$$\frac{d}{dt} m(f_{\theta_t}) = \mathscr{E}(m(f_{\theta_t}), R_t).$$

We refer to the pair of a learning process and a metric m as a learning dynamics.

[a] For instance, in the limit of infinitesimally small learning steps.

Definition:

Definition 11. ***Learning Progress*** *- Given learning dynamics $\mathscr{E}$ and an associated metric m, we say that a model exhibits m-learning progress if and only if there exists $T > 0$ such that:*

$$\forall t \geq T, \quad m(f_{\theta_t}) < m(f_{\theta_0}).$$

A key observation follows: it is experimentally impossible to definitively verify whether a model has achieved (or failed to achieve) learning progress. One can only make hypotheses based on empirical observations.

Remark:

Remark 5. *Suppose there exists a metric m and a time $T > 0$ such that $\forall t \geq T$, $m(f_{\theta_t}) > m(f_{\theta_0})$. Then the model is said to have achieved $(-m)$-learning progress.*

The notion of learning progress is not intended to evaluate a model's task-specific performance, but rather to capture the extent to which the model acquires and retains information during training.

Definition:

Definition 12. ***Stable Learning*** *- Given learning dynamics and an associated metric m, we say the dynamics exhibit m-stable learning if there exists a constant C such that:*

$$|m(f_{\theta_{t+1}}) - m(f_{\theta_t})| < C, \quad \forall t \geq 0.$$

Convergent Learning *- The learning dynamics is said to be m-convergent if:*

$$|m(f_{\theta_{t+1}}) - m(f_{\theta_t})| \to 0 \quad \text{as } t \to \infty.$$

Note that this notion of stability is unrelated to the classical definition of stability in machine learning (e.g., generalization-based uniform stability). To build some intuition regarding the limitations of the aforementioned definitions, consider

that the convergence of learning -as defined through interpretability metrics- may be strongly correlated with training convergence.[18] This question will be addressed later. However, we emphasize that convergence with respect to an interpretability metric does not necessarily imply convergence in the underlying learned representations. Such behavior may be entirely attributable to the training dynamics' inertia, rather than meaningful stabilization of internal model knowledge.

3.2 On the Concept Framework

Concept-based interpretability represents one of the most developed areas in mechanistic interpretability. Indeed, concept retrieval is often performed through automated methods that analyze internal activations, making such techniques both accessible for large language models (LLMs) and highly insightful. In this section, we aim to outline a general framework for these methods.

Before proceeding, it is important to acknowledge that there is no consensus on what constitutes a *concept*. The term itself appears to be gradually evolving toward the broader notion of a *feature*, reflecting the flexibility and richness of these interpretability approaches. Nevertheless, for the sake of consistency, we will continue to refer to them as *concepts*.

Furthermore, concept-based methods are commonly distinguished as either *supervised* or *unsupervised*[19] [PCP+23].

Supervised concepts are typically obtained through human annotation or label-driven training, and the resulting representations are usually human-interpretable. Even if the underlying method is more complex, the final outputs are intended to align with human semantics.

Unsupervised concepts, on the other hand, are extracted through clustering or dimensionality reduction techniques applied to internal representations. These concepts often do not correspond to human-intuitive categories, but rather reflect the model's internal latent space-often incomprehensible to humans, especially in the case of large models.

Definition:

Definition 13. ***Concept Extractor*** - *Given a decomposable model $f = g \circ h$ and an input $x \in \mathcal{X}$, we define a* concept extractor *as a function $\phi : \mathcal{H} \to \mathcal{C} \subset \mathbb{R}^m$, where $\mathcal{C}$ is the latent concept space and ϕ is non-constant. In case of a θ-parameter dependence, we write $\Phi_f(\theta, x) = \phi(h_\theta(x))$.*

It is important to note that this definition is intentionally general and designed to offer flexibility. For instance, concepts can be associated with human-defined labels: one might define specific vectors to represent particular concepts, such as $\mathbf{e}_i = (0, \ldots, 1, \ldots, 0)^\top$ encoding the concept "shape".

Let us now more precisely define what we will refer to as a *concept*-also referred to interchangeably as a *feature*.

[18] It is nontrivial to select a specific notion of convergence to characterize training. One might refer to loss convergence, e.g., $|\mathcal{L}(f_{\theta_{t+1}}(x), y) - \mathcal{L}(f_{\theta_t}(x), y)| \xrightarrow[t\to\infty]{} 0$, or alternatively to convergence in parameter space, e.g., $\|\theta_{t+1} - \theta_t\| \xrightarrow[t\to\infty]{} 0$ for a chosen norm $\|\cdot\|$.

[19] The terms "supervised" and "unsupervised" here refer to task settings. A supervised task is one for which humans provide explicit objectives, such as labeled data, and the model is expected to replicate a human-like understanding of the data distribution. In contrast, unsupervised tasks require minimal human intervention and are designed under the assumption that the data contains inherent structure. For example, a classification task that distinguishes images of cats and dogs using labeled data is supervised. If labels are removed and the model instead attempts to infer structure through techniques like clustering, the task becomes unsupervised.

Definition:

Definition 14. ***Mechanistic Concept*** *- Given a fixed decomposable model $f = g \circ h$ and an input $x \in \mathcal{X}$, we say that a vector $w \in \mathbb{R}^m$ is a* concept *if and only if there exists a concept extractor ϕ such that*

$$w = \phi(h(x)).$$

Note that we specifically refer here to **mechanistic concepts**, which constitute the main focus of our study. The broader field of concept-based methods has roots in category theory and allows for more elaborate and abstract definitions of concepts.

Remark:

Remark 6. *For any $w \in \mathbb{R}^m$, we can build a valid concept extractor such that for fixed h and x, $\phi(h(x)) = w$. Therefore, any vector in $\mathbb{R}^m$ qualifies as a mechanistic concept under this definition.*

This observation arises from the flexible nature of the concept extractor definition. Such flexibility makes this framework highly adaptable and easy to manipulate. However, it also implies that deriving general or theoretical results may be challenging. Later in this work, we will introduce additional assumptions on the concept extractor in order to enable more rigorous analysis.

Remark:

Remark 7. *For better scalability of the above definitions, we will assume that concept extractors can be extended to return multiple concepts simultaneously. Formally, we allow extractors to output a matrix $W \in \mathbb{R}^{m \times r}$ containing r concept vectors, each of dimension m. We denote this multi-concept extractor as $\phi^r(h(x)) = \Phi_f^r(\theta, x) = W$.*

3.3 On the General Framework

The general framework developed in this work is structured as follows. Given a learning process denoted by $\mathscr{L} = (f_*, \mathcal{A}_L, \theta_0)$ and a r-concept extractor Φ_f^r, we will build a metric in order to derive a leaning dynamics. Then, we will try to asses the learning dynamics properties theoretically and empirically.

We begin by recalling some essential definitions related to distances and metrics:

Definition:

Definition 15. ***Distance*** *- Let X be a non-empty set. A function $d : X \times X \to \mathbb{R}$ is called a* distance function *if it satisfies the following properties:*

1. ***Non-negativity:*** $\forall\, x, y \in X,\ d(x, y) \geq 0.$
2. ***Identity of indiscernibles:*** $d(x, x) = 0 \Leftrightarrow x = y.$
3. ***Triangle inequality:*** $\forall\, x, y, z \in X,\ d(x, z) \leq d(x, y) + d(y, z).$

Algorithm 1: AdamW Optimizer [LH19] & PyTorch

input : η: learning rate; $\beta_1, \beta_2 \in [0, 1)$: exponential decay rates for moment estimates;
ϵ: small constant for numerical stability;
λ: weight decay coefficient;
θ_0: initial parameters;
$L(\theta)$: loss function.

output: Trained parameters θ_T

Initialize $m_0 \leftarrow 0$, $v_0 \leftarrow 0$, $t \leftarrow 0$

for *each iteration* $t = 1$ *to* T **do**

- Compute gradient: $g_t \leftarrow \nabla_\theta L(\theta_{t-1})$
- Update biased first moment estimate: $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
- Update biased second moment estimate: $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
- Compute bias-corrected first moment: $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$
- Compute bias-corrected second moment: $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$
- Parameter update: $\theta_t \leftarrow \theta_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_{t-1} \right)$

Definition:

Definition 16. ***Metric*** *- Let X be a non-empty set, and let d be a distance on X. Given a reference element $y \in X$, we define the* metric *$m : X \to \mathbb{R}$ as the function $m(x) = d(x, y)$.*

Throughout this work, we will focus on evaluating learning dynamics using the EuroBERT model family as our primary learning system [BGBA+25].

EuroBERT[20] is a family of multilingual encoders trained on a wide range of European languages.

The term *encoder* here refers to a function that transforms an input token sequence into a contextual embedding matrix. Formally, let $x = (x_1, x_2, \ldots, x_n)$ be a sequence of tokens. An encoder computes contextual representations

$$\mathbf{H} = f_{\text{enc}}(x), \quad \mathbf{H} \in \mathbb{R}^{n \times d},$$

where each row $\mathbf{h}_i \in \mathbb{R}^d$ corresponds to the contextual embedding of token x_i.

EuroBERT models are built with Transformer layers. Currently, the family contains three models: EuroBERT-210M, EuroBERT-610M, and EuroBERT-2.1B. The numbers indicate the parameter size: the smallest has 210 million parameters, the next 610 million, and the largest 2.1 billion. Each of these models has been trained on the MLM task (see Section 2.2 for details) for approximately 500,000 steps, with checkpoints available every 10,000 steps.

The optimization algorithm used is AdamW [LH19] (see Algorithm 1). Note, however, that the update rule combines both the optimizer algorithm and the MLM training objective:

$$\min_{\theta \in \Theta} \mathbb{E}_{(x,y) \sim \mathcal{D}_{MLM}} \left[\mathcal{L}_{CE}(f(\theta; x), y) \right]. \tag{MLM}$$

[20] The name *EuroBERT* comes from the famous family of encoder models BERT, which first introduced Transformers in a bidirectional encoder setting [DCLT19]. EuroBERT belongs to this family but is specifically trained on multiple European languages (not only English), hence the "Euro" prefix.

4 Mechanistic Overview

In this part, we will aim to make an overview of the different methods from mechanistic interpretability we will use in order to extract concepts. From their, we shall derive some interesting property for those methods. This is not a exhaustive presentation of all methods in the field, but a short presentation in order to allow for property study.

4.1 Unsupervised Extractor General

In this section, we present how most unsupervised concept extractors (UCE) work in mechanistic interpretability. To our knowledge no proper overview with general mathematical framework has been constructed on this peculiar subject. Given an error function $\mathcal{E} : \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}_+$, an activation matrix $A \in \mathbb{R}^{n\times m}$, and a fixed number of concepts to extract $r \in \mathbb{N}$, we aim to solve:

$$\min_{U\in\mathcal{J}\subset\mathbb{R}^{n\times r},\, W\in\mathcal{K}\subset\mathbb{R}^{m\times r}} \mathcal{E}(A, UW^T). \tag{UCE}$$

The matrix U is called the *encoded activation* or *concept coefficients*, while W is the *concept dictionary*[21]. The main idea is that the activation matrix A results from concept vectors being linearly combined through weights from the concept coefficients. In the general case, we add constraints to the spaces of U and W. An unsupervised concept extractor can therefore be defined as follows:

Definition:

Definition 17. ***Unsupervised Concept Extractor*** - *Given an error function $\mathcal{E} : \mathbb{R}^{n\times m} \times \mathbb{R}^{n\times m} \longrightarrow \mathbb{R}_+$ and constrained sets*[a] *$\mathcal{K} \subset \mathbb{R}^{m\times r}$ and $\mathcal{J} \subset \mathbb{R}^{n\times r}$, let $f = g \circ h$ be a decomposable model and $x \in \mathcal{X}$ an input. We define an* unsupervised concept extractor *as a function $\phi : \mathcal{H} \to \mathcal{C} \subset \mathbb{R}^m$, where $\mathcal{C}$ is the latent concept space, such that*[b]

$$\phi(h_\theta(x)) \in \arg\min_{W\in\mathcal{K}} \left(\min_{U\in\mathcal{J}} \mathcal{E}\big(h_\theta(x), UW^\top\big) \right).$$

[a] Note that $\mathcal{K}$ and $\mathcal{J}$ may depend on $h_\theta(x)$, which we will then write : $\mathcal{K}(h_\theta(x))$ and $\mathcal{J}(h_\theta(x))$.
[b] Note that we can assume that the concept extractor also return U. At least for computing, we will sometime note $(U, W) = \phi(h_\theta(x))$.

There is little guarantee that the extractor is unique. In this section, we reflect on the reasons for this and explain why it is not such a major problem in our case.

First, let X^* denote the solution to the problem:

$$\mathcal{M}_r := \{X \in \mathbb{R}^{n\times m} : \operatorname{rank}(X) \le r\}, \qquad \min_{X\in\mathcal{M}_r} \mathcal{E}(A, X). \tag{1}$$

Then, problems (UCE) and (1) are equivalent, assuming we write $X^* = U^*(W^*)^T$ (see Property 3). However, depending on $\mathcal{E}$ and A, this problem can admit multiple solutions X^* (implying multiple extractors). This mostly comes from the fact that we are selecting non-full-rank matrices.

A simple example of problem (1) would be:

$$\min_{X\in\mathcal{M}_r} \|I_n - X\|_1.$$

[21] It is not exactly a dictionary learning problem. Some of the concept extractors are indeed dictionary learning solutions but not all.

For simplicity, assume $n = m$. Then I_n is the identity matrix of rank n. If $r < n$, then all matrices X with r ones on their diagonal (any permutation allowed) and zeros elsewhere would be optimal solutions.

While uniqueness is not a problem for post-hoc mechanistic methods[22], it is an issue in our specific case. Indeed, if uniqueness is not guaranteed and the optimum selector is randomized, then the concept space representation we build at each learning step cannot be consistently compared. This is because we could be comparing different solutions rather than the true evolution of the concept space. This is something we will study later on.

4.2 Non-Negative Matrix Factorization (NMF)

Non-Negative Matrix Factorization (NMF) is a matrix decomposition method whose modern form was introduced by Daniel Lee and Sebastian Seung [LS99]. The original objective is to solve the following problem: given a matrix $A \in \mathbb{R}_+^{n \times m}$ and a rank value $r \in \mathbb{N}$, we aim to solve

$$\min_{U \in \mathbb{R}^{n \times r},\, W \in \mathbb{R}^{m \times r}} \|A - UW^\top\|_F^2 \quad \text{subject to } U \geq 0,\ W \geq 0. \tag{NMF}$$

It is important to note that this method is rather restrictive on the activation matrix A: indeed, A must be non-negative. The intuition behind NMF is to identify which features were *added together* (rather than subtracted) to produce the matrix. In our case, these features correspond to concept vectors.

Now, let us define the NMF extractor we will be using:

Definition:

Definition 18. ***NMF Extractor*** *- An NMF extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E} : \mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J} : \mathcal{J} = \mathbb{R}_+^{n \times r}$
- $\mathcal{K} : \mathcal{K} = \mathbb{R}_+^{m \times r}$

4.3 Convex NMF

Convex NMF is a specific variant of NMF. The form used in mechanistic interpretability comes from a method introduced by Chris Ding, Tao Li, and Michael I. Jordan [DLJ10]. The issue with classic NMF is that it sometimes generates "ghost" features that are not actually present in the data. To prevent this, the decomposition is constrained so that the outputs lie within the convex hull of the data. This leads to the following problem, given $A \in \mathbb{R}_+^{n \times m}$:

$$\min_{U \in \mathbb{R}^{n \times r},\, W \in \mathbb{R}^{m \times r},\, H \in \mathbb{R}^{m \times r}} \|A - UW^\top\|_F^2 \quad \text{subject to} \quad U = AH,\ H \geq 0,\ \mathbf{I}_m^\top H = \mathbf{I}_r^\top,\ W \geq 0. \tag{cNMF}$$

Here, U is a projection over the space generated by A, the activation matrix. This forces the method to extract features directly from A. We can then define the convex NMF extractor as follows:

[22] It is not considered an issue because those methods do not give specific signification to concept vector. Indeed, they mostly process them in order to retrieve something : Wether it is gradient importance, visualization comparison or anything else.

Definition:

Definition 19. ***Convex NMF Extractor*** *- A convex NMF extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E}$: $\mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J}(A)$: $\mathcal{J}(A) = \{U \in \mathbb{R}_+^{n\times r} \mid U = AH,\ H \geq 0,\ \mathbf{I}_m^\top H = \mathbf{I}_r^\top\}$
- $\mathcal{K}$: $\mathcal{K} = \mathbb{R}_+^{m\times r}$

4.4 Semi NMF

Semi NMF is also a variant of NMF from the same authors than convex NMF [DLJ10]. It is mainly the same as NMF but accepting negative value in A thanks to the fact that the coefficient space $\mathcal{K}$ is uncontrained, this give us the following definition :

Definition:

Definition 20. ***Semi NMF Extractor*** *- A semi-NMF extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E}$: $\mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J}$: $\mathcal{J} = \mathbb{R}^{n\times r}$
- $\mathcal{K}$: $\mathcal{K} = \mathbb{R}_+^{m\times r}$

4.5 Singular Value Decomposition (SVD)

Singular Value Decomposition (SVD) is a fundamental matrix factorization method whose origins trace back to work by the Italian mathematician Eugenio Beltrami (1873) and the French mathematician Camille Jordan (1874). The original objective is to solve the following problem: given a matrix $A \in \mathbb{R}^{n\times m}$ and a target rank $r \in \mathbb{N}$, we aim to solve

$$\min_{U\in\mathbb{R}^{n\times r},\, V\in\mathbb{R}^{m\times r},\, \Sigma\in\mathbb{R}^{r\times r}} \|A - U\Sigma V^\top\|_F^2 \quad \text{subject to } U^\top U = I_r,\ V^\top V = I_r,\ \Sigma \text{ diagonal},\ \Sigma \geq 0. \tag{SVD}$$

Unlike NMF, this formulation imposes no sign restrictions on A: the SVD exists for any real (or complex) matrix. The intuition behind SVD is that it identifies the *directions of maximum variance and correlation* in the data, decomposing A into orthogonal bases U and V scaled by singular values in Σ. In our case, these directions correspond to the most informative latent factors that best approximate the original matrix. To retrieve explicitly our encoded activation and concepts bank, we will need to define $\Sigma^{1/2} = diag((\sqrt{\Sigma_{ii}})_{i\in\mathbb{N}})$. It verifies that $\left(\Sigma^{1/2}\right)^2 = \Sigma$ and we then have $\hat{U} = U\Sigma^{1/2}$ and $\hat{V} = \Sigma^{1/2}V$. If we denote $\Sigma(A)$ the singular value matrix obtained through SVD, we can define the SVD extractor :

Definition:

Definition 21. ***SVD Extractor*** - *An SVD extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E} : \mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J}(A) : \mathcal{J}(A) = \{U \in \mathbb{R}^{n\times r} \mid U^\top U = \Sigma(A)\}$
- $\mathcal{K}(A) : \mathcal{K}(A) = \{V \in \mathbb{R}^{m\times r} \mid V^\top V = \Sigma(A)\}$

Note that the solution to PCA is tightly linked to SVD solution. Indeed, both represent the direction of maximum variance in the data. The difference lies on the way to handle the singular value matrix.

4.6 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) was first introduced by Karl Pearson [Pea01]. It was mostly designed to select proper dimension frame in order to treat high dimensional data. A first mention of PCA in a form of dictionary learning problem is made by Bruno Olshausen and David Field [OF96]. It is possible to prove that a centered PCA solution correspond to the solution of the following problem (See Proof **??**) :

$$\min_{U\in\mathbb{R}^{n\times r},\, W\in\mathcal{O}} \|A - UW^\top\|_F^2, \quad \mathcal{O} = \{D \in \mathbb{R}^{m\times r} : \; D^\top D = I_r\}, \quad \mathbf{1}_n^\top U = \mathbf{0}_{1,r}. \tag{PCA}$$

Which allow us to write the definition for PCA extractor as :

Definition:

Definition 22. ***PCA Extractor*** - *An PCA extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E} : \mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J} : \mathcal{J} = \{U \in \mathbb{R}^{n\times r} : \mathbf{1}_n^\top U = \mathbf{0}_{1,r}\}$
- $\mathcal{K} : \mathcal{K} = \{W \in \mathbb{R}^{m\times r} : \; W^\top W = I_r\}$

4.7 Sparse PCA

Sparse PCA is a ℓ_1-norm penalized PCA (under acceptable assumptions, with Property 4) we define :

Definition:

Definition 23. ***Sparse PCA Extractor*** - *Given a fixed $\tau > 0$, a sparse PCA extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E} : \mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J} : \mathcal{J} = \{U \in \mathbb{R}^{n\times r} : \mathbf{1}_n^\top U = \mathbf{0}_{1,r}\,, \|U\|_1 \leq \tau\}$
- $\mathcal{K} : \mathcal{K} = \{W \in \mathbb{R}^{m\times r} : \; W^\top W = I_r\}$

4.8 Independent Component Analysis

Independent Component Analysis (ICA) was first introduced by A. Hyvärinen and E. Oja [HO00]. ICA was introduced to enforce even more statistical independence between obtained component. Indeed, PCA enforce non-correlation between component but this is only second order statistics and does not accurately shows independence of input in non-gaussian sources for example. In our situation, ICA is less interpretable because of its restricted solution space for which it is more difficult to draw conclusion but it give more accurate sources to understand the evolution of the latent space, indeed the representation are more robust assuring possible uniqueness for each steps. Introducing the following object, we can write the ICA problem in the form of our unsupervised extractor :

Definition:

Definition 24. ***Centered data matrix*** *- Let $X \in \mathbb{R}^{n\times m}$ be a data matrix whose rows are observations. The* centered data matrix *(also written $\widehat{X}$ or $\widetilde{X}$) is (i.e. each column of $\widehat{X}$ has zero empirical mean.)*

$$\widehat{X} \;:=\; X - \mathbf{1}_n \bar{x}^\top \;=\; \Big(X_{ij} - \frac{1}{n}\sum_{t=1}^{n} X_{tj}\Big)_{i=1,\dots,n;\; j=1,\dots,m}.$$

Definition:

Definition 25. ***Empirical Approximated Negentropy*** *- Let y be an observed vector $y = (y_1, \dots, y_n)^\top$), negentropy is estimated using a nonlinear function $G : \mathbb{R} \to \mathbb{R}$. With empirical approximation :*

$$\widehat{J}_G(y) \;:=\; \left| \frac{1}{n}\sum_{i=1}^{n} G\Big(\frac{y_i}{\widehat{\sigma}_y}\Big) \;-\; \mathbb{E}\big[G(\nu)\big] \right|,$$

where $\widehat{\sigma}_y^2 = \frac{1}{n}\sum_{i=1}^{n} y_i^2$ and $\nu \sim \mathcal{N}(0,1)$. Typical choices for G used in practice are

$$G(u) = \log\cosh(au), \qquad G(u) = 1 - \exp(-u^2/2), \qquad G(u) = u^4,$$

Definition:

Definition 26. ***Whitening (sphering) matrix K for centered data*** *- Let $X \in \mathbb{R}^{n\times m}$ be a data matrix. A (full)* whitening matrix *$K \in \mathbb{R}^{m\times m}$ is any matrix satisfying*

$$\frac{1}{n}\,(\widehat{X}K)^\top(\widehat{X}K) \;=\; I_m,$$

i.e. the whitened data $Z := \widehat{X}K$ has identity empirical covariance.

Remark:

Remark 8. ***Rank-reduced / truncated whitening (practical).*** *If one intends to retain only the leading r dimensions ($r \leq \operatorname{rank}(\widehat{\Sigma})$), let $V_r \in \mathbb{R}^{m\times r}$ be the matrix of the top-r eigenvectors and $\Lambda_r \in \mathbb{R}^{r\times r}$ the corresponding eigenvalues. A compact (reduced) whitening mapping $K_r \in \mathbb{R}^{m\times r}$ is then*

$$K_r \;=\; V_r \Lambda_r^{-1/2},$$

and the reduced whitened data $Z_r := \widehat{X} K_r \in \mathbb{R}^{n\times r}$ satisfies

$$\frac{1}{n} Z_r^\top Z_r \;=\; I_r.$$

Equivalently, if $\widehat{X} = U\Sigma V^\top$ is the thin SVD, one may take $K_r = V_r \Sigma_r^{-1}$ so that $Z_r = U_r \Sigma_r V_r^\top V_r \Sigma_r^{-1} = U_r$.

Let $X \in \mathbb{R}^{n\times m}$ be a data matrix and given $\boldsymbol{\tau} \in \mathbb{R}_+^n$, let us define our constrained space :

$$\mathcal{U}_{\mathrm{ICA}}(X, \boldsymbol{\tau}) := \{\, U \in \mathbb{R}^{n\times r} :\; \exists Q \in \mathcal{O}(r),\; U = \widehat{X} K_r Q,\; \mathbf{1}_n^\top U = 0,\; \widehat{J}_G(u_j) \geq \tau_j \;\forall j \}.$$

Giving us the following constrained minimization problem :

$$\min_{U \in \mathcal{U}_{\mathrm{ICA}}(A, \boldsymbol{\tau}),\; W \in \mathbb{R}^{m\times r}} \|A - UW^\top\|_F^2, \tag{ICA}$$

Definition:

Definition 27. ***ICA Extractor*** *- Given a fixed $\tau > 0$, an ICA extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E}$ *:* $\mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J}(A)$ *:* $\mathcal{J}(A) = \mathcal{U}_{\mathrm{ICA}}(A, \boldsymbol{\tau})$
- $\mathcal{K}$ *:* $\mathcal{K} = \mathbb{R}^{m\times r}$

4.9 K-Means

K-Means is a unsupervised methods used in a wide variety of fields, thus its form vary depending on some subject. This algorithm has found multiple application since its first proper definition in 1967 [Mac67]. Rewriting K-Means for our case give us the following problem :

$$\min_{U \in \{0,1\}^{n\times r},\, W \in \mathbb{R}^{m\times r}} \|X - UW^\top\|_F^2 \quad \text{subject to } U\mathbf{1}_r = \mathbf{1}_n. \tag{KMeans}$$

Which allow us to define the following extractor :

Definition:

Definition 28. ***kMeans Extractor*** - *A k-Means extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E} : \mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J} : \mathcal{J} = \{U \in \{0,1\}^{n \times r},\ U\mathbf{1}_r = \mathbf{1}_n\}$
- $\mathcal{K} : \mathcal{K} = \mathbb{R}^{m \times r}$

4.10 Dictionary Learning

Dictionary Learning (DictL) is by definition the purest form of concept extractor to us. It definitely overlap with unsupervised concept extractor but is not exactly the same. Indeed, some method cannot be purely defined with dictionary learning framework. Here, we will use the following dictionary learning problem (See Problem DictL) but this method creator did intend for more general use [MBPS09]. Given $\lambda > 0$:

$$\min_{U \in \mathbb{R}^{n \times r},\, W \in \mathbb{R}^{m \times r}} \|A - UW^\top\|_F^2 \ + \ \lambda \|U\|_1 \tag{DictL}$$

Which we can rewrite under acceptable assumptions (See Property 4) as :

Definition:

Definition 29. ***DictionnaryLearning Extractor*** - *Given $\tau > 0$, a dictionnary learning extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E} : \mathcal{E}(A, B) = \|A - B\|_F^2$
- $\mathcal{J} : \mathcal{J} = \{U \in \mathbb{R}^{n \times r},\ \|U\|_1 \leq \tau\}$
- $\mathcal{K} : \mathcal{K} = \mathbb{R}^{m \times r}$

4.11 Sparse Auto-Encoder

Sparse Auto-Encoder (SAE) is the standard unsupervised method for MI [Ng11]. It has various application and multiple form. It is also part of the method that cannot be rewritten as explicit dictionnary learning (even if they share a lot in common). SAE is a method that train an auto-encoder which is a neural network composed of an encoder and a decoder. The training process solve the following problem :

$$\min_{\theta \in \Theta} \|A - D_\theta \circ E_\theta(A)\|^2 \ + \ \lambda \|P_{D_*}(\theta)\|_1 \tag{fullSAE}$$

With P_{D_*} being a projection on the parameter of the decoder.

The norm is purposely not defined because any norm would permit to do such a reconstruction loss. In our specific case of Mechanistic Interpretability, we want an **interpretable** output for our method. Thus, in our case the decoder part will be a matrix with sparse constraint. And the encoder part will return an encoded matrix. First, we will set the

constrained space for the encoder. Given a Model Architecture $E_* \in \mathcal{NN}$, we define :

$$\mathcal{J}_{enc}(A) = \{U \in \mathbb{R}^{n \times r} \mid \exists \theta \in \Theta,\ U = E_\theta(A)\}$$

This allow us to rewrite this problem as follow (we will use the norm 2 which is most used in pre-implemented SAE):

$$\min_{U \in \mathcal{J}_{enc}(A),\ W \in \mathbb{R}^{m \times r}} \|A - UW^\top\|_2^2 \ + \ \lambda \|W\|_1 \tag{SAE}$$

Which we can rewrite as the following extractor under acceptable assumptions (See Property 4):

Definition:

Definition 30. ***SAE Extractor*** - *Given $\tau > 0$, a SAE extractor is an unsupervised extractor (Def. 17) such that:*

- $\mathcal{E}$: $\mathcal{E}(A, B) = \|A - B\|_2^2$
- $\mathcal{J}$: $\mathcal{J}(A) = \mathcal{J}_{enc}(A)$
- $\mathcal{K}$: $\mathcal{K} = \{W \in \mathbb{R}^{m \times r},\ \|W\|_1 \leq \tau\}$

4.12 Properties

4.12.1 Families of Extractors

The extractors defined above can be grouped into families based on the principles through which they obtain their.

NMF Family: A first and rather intuitive family of extractors is based on Non-Negative Matrix Factorization. This includes the standard (vanilla) NMF, semi-NMF, and Convex NMF. Each of these methods aims to extract *positive* concepts, meaning that activations are expressed as additive combinations of meaningful component vectors. The core intuition behind this family is that the activation matrix is primarily the sum of positive concept vectors - that is, features that are truly present in the decomposition, rather than artifacts of cancellation between positive and negative terms.

SVD Family: The Singular Value Decomposition family relies on singular values to capture structure within the data. When the data are centered, singular values correspond to the distribution of variance, making PCA and Sparse PCA natural statistical extensions of SVD. Their primary goal is to identify the directions along which activations vary the most. Under the assumption that the activation distribution (or manifold) is approximately linear (at least locally), such decompositions can provide valuable insight into the underlying geometry of the activation space.

Sparse Learning Family: The sparse learning-based family includes methods such as Dictionary Learning, k-Means, and Sparse Autoencoders (SAE). Their shared objective is to extract from the activations a compact set of representative features. Sparsity constraints play a central role here: by enforcing sparsity on the concept coefficients, these methods ensure that only the most relevant features emerge. k-Means can be viewed as a special case of this family, as it effectively hard-codes sparsity by assigning each activation to a single cluster, whereas Dictionary Learning and SAE use penalization to achieve similar effects. Sparse PCA does not rely as much as these methods on the learning process, this is why it was not included in this family. Finally, although SAE could arguably form a family of its own-given the large number of existing variants-we will not delve too deeply into it, as it fails to produce coherent learning dynamics within our analytical

framework.

Other possible family - Statistic Family: The ICA methods is by design unrelated to each of the family described above. However, ICA is a decomposition that use high order statistic to enforce independence on its component. While PCA use in fact second order statistics (correlation) to enforce independence. Thus, it would not be shocking to class ICA, PCA and sparse PCA into the same family of statistic related methods. However, doing so won't account for the underlying structure of those methods that are highly different (Just like SVD, PCA is but a linear global fitting to some subspaces, ICA has no clear such spatial behavior).

4.12.2 Solution Structure

One of the fundamental challenges with decomposition methods is that their solutions are often *non-unique*. Even when multiple valid solutions exist, interesting patterns or structures tend to emerge within the space of all possible factorizations. To formalize this, we define several equivalence relations that capture common transformation symmetries between solutions.

Definition:

Definition 31. ***Permutation Equivalence*** - *Let $U \in \mathbb{R}^{n\times r}$ and $W \in \mathbb{R}^{m\times r}$ be factor matrices. We define the permutation equivalence relation $\sim_{\text{Perm}}$ on pairs (U, W) by*

$$(U, W) \sim_{\text{Perm}} (U', W') \iff \exists P \in \mathbb{R}^{r\times r} \text{ permutation matrix such that } U' = UP \text{ and } W' = WP.$$

Definition:

Definition 32. ***Scaling Equivalence*** - *Let $U \in \mathbb{R}^{n\times r}$ and $W \in \mathbb{R}^{m\times r}$ be factor matrices. We define the scaling equivalence relation $\sim_{\text{Scale}}$ on pairs (U, W) by*

$$(U, W) \sim_{\text{Scale}} (U', W') \iff \exists D \text{ diagonal with } D_{ii} \neq 0 \text{ such that } U' = UD \text{ and } W' = WD^{-1}.$$

Definition:

Definition 33. ***Sign-Flip Equivalence*** - *Let $U \in \mathbb{R}^{n\times r}$ and $W \in \mathbb{R}^{m\times r}$ be factor matrices. We define the sign-flip equivalence relation $\sim_{\text{Sign}}$ on pairs (U, W) by*

$$(U, W) \sim_{\text{Sign}} (U', W') \iff \exists D \text{ diagonal with } D_{ii} \in \{-1, 1\} \text{ such that } U' = UD \text{ and } W' = WD.$$

Definition:

Definition 34. ***Orthogonal Rotation Equivalence*** - *Let $U \in \mathbb{R}^{n\times r}$ and $W \in \mathbb{R}^{m\times r}$ be factor matrices. We define the orthogonal rotation equivalence relation $\sim_{\text{Rot}}$ on pairs (U, W) by*

$$(U, W) \sim_{\text{Rot}} (U', W') \iff \exists Q \in \mathcal{O}(r) \text{ such that } U' = UQ \text{ and } W' = WQ.$$

These equivalence relations describe how different decompositions can represent essentially the same factorization, up to transformations that preserve the structure of the reconstruction $A \approx UW^\top$. Let us now examine how these equivalences manifest across different families of decomposition methods.

SVD Family: The SVD family is among the best understood. Given two solution pairs (U, W) and (U', W') of a fixed matrix A, we have the following properties:[23]

- If A has strictly distinct eigenvalues, then (U, W) is unique up to **sign-flip equivalence**.
- If A contains repeated eigenvalues, then (U, W) is unique up to both **sign-flip** and **orthogonal rotation equivalence**.

PCA, being a constrained form of SVD, exhibits the same equivalence structure on (U, W), although the loadings W are partially restricted by normalization conditions.

NMF Family: NMF is known to be an *ill-posed* problem - it can produce pathological solutions with no structural relation between them [PDV16]. Semi-NMF and Convex NMF were introduced specifically to mitigate these issues [DLJ10]. Convex NMF, in particular, has been shown empirically to yield unique solutions up to **permutation equivalence** [OMLV+12].

Sparse Learning Family: Sparse Autoencoders (SAE) and Dictionary Learning belong to a highly flexible but often unstable class of methods. For instance, given any invertible matrix $M \in \mathbb{R}^{r\times r}$ with $\|M\|_1 = \epsilon \leq \tau$, a valid solution (U, W) implies that $(UM, WM^{-\top})$ is also a solution - meaning that the representation space can easily degenerate if ϵ is chosen arbitrarily small. For k-Means, non-uniqueness arises from local minima, though any **permutation** of cluster labels still yields an equivalent valid solution [CCR24].

ICA Family: Under certain strict assumptions, Independent Component Analysis (ICA) can achieve uniqueness up to **orthogonal rotation equivalence** [HP99]. However, in practice, these assumptions rarely hold, and ICA may also produce pathological or unstable solutions.

Decomposition Family	$\sim_{\text{Perm}}$	$\sim_{\text{Scale}}$	$\sim_{\text{Sign}}$	$\sim_{\text{Rot}}$
SVD / PCA	×	×	✓	✓ (if repeated eigenvalues)
NMF	+	+	+	+
Semi-NMF / Convex NMF	✓	×	×	×
Dictionary Learning	+	+	+	+
k-Means	✓	×	×	×
ICA	+	+	+	+/✓ (under assumptions)

Table 1: Summary of equivalence relationships across decomposition families. A checkmark (✓) indicates that solutions are equivalent up to the corresponding transformation, a cross (×) indicates absence of that equivalence in general and a (+) indicates that in solution are pathological implying that we may have any of the above relationship **(Wich means that we do not know the structure of the solution)**.

5 Observing Representation

Now that we have presented the complete framework inside which we will place our study. We can define the general process that we will perform (See Algorithm 2). As detailed before, the output that we retrieve is a measure sequence. This allow us to formulate a learning dynamics (See Def 10).

[23] See Cornell Notes and Math StackExchange.

Algorithm 2: Compute concept representations across checkpoints and similarity to final-step concepts

input : x: input dataset (batch size $B = N_{samples} \times N_{tokens}$);
f_θ: model architecture (decomposable model / mapping $(x, \theta) \mapsto h_\theta(x)$);
$\{\theta_t\}_{t=1}^{T}$: snapshot parameters at timesteps $1, \ldots, T$;
$\{P_k\}_{k=1}^{K}$: list of projectors where $P_k(f_\theta, x)$ returns activation(s) of the k-th layer;
ϕ: concept extractor, $\phi(h) \mapsto (U, W)$;
m: similarity metric $m(U_1, W_1, U_2, W_2) \in \mathbb{R}_+$.

output: Lists $\mathcal{U} = [U_1, \ldots, U_T]$, $\mathcal{W} = [W_1, \ldots, W_T]$, and similarities $\mathcal{S} = [s_1, \ldots, s_T]$.

Initialize empty lists $\mathcal{U} \leftarrow []$, $\mathcal{W} \leftarrow []$, $\mathcal{S} \leftarrow []$

Compute list (for each layers) of concatenated batch of activations (in $\mathbb{R}^{B \times d_{hidden}}$) for final checkpoint:

$$(H_T^{B,k})_{k=1}^{K} \leftarrow \left(P_k(f_{\theta_T}, x)\right)_{k=1}^{K}$$

Retrieve reference concepts: $(U_T^k, W_T^k)_{k=1}^{K} \leftarrow \left(\phi(H_T^{B,k})\right)_{k=1}^{K}$

Append $(U_T^k)_{k=1}^{K}$ to $\mathcal{U}$ and $(W_T^k)_{k=1}^{K}$ to $\mathcal{W}$ (as the last element)

for *each* $t = 1$ **to** T **do**

Compute activations at checkpoint t:

$$(H_t^{B,k})_{k=1}^{K} \leftarrow \left(P_k(f_{\theta_t}, x)\right)_{k=1}^{K}$$

Extract concepts: $(U_t^k, W_t^k)_{k=1}^{K} \leftarrow \left(\phi(H_t^{B,k})\right)_{k=1}^{K}$

Append $(U_t^k)_{k=1}^{K}$ to $\mathcal{U}$ and $(W_t^k)_{k=1}^{K}$ to $\mathcal{W}$

Compute similarity to reference:

$$(s_t^k)_{k=1}^{K} \leftarrow \left(m(U_t^k, W_t^k, U_T^k, W_T^k)\right)_{k=1}^{K}$$

Append $(s_t^k)_{k=1}^{K}$ to $\mathcal{S}$

return $\mathcal{U}$, $\mathcal{W}$, $\mathcal{S}$

5.1 Empirical Learning Dynamics

We now construct the empirical learning dynamics using Algorithm 2. To do so, we first define a set of norms that will serve as metrics to assess concept space convergence. These norm-based metrics are formally introduced in Section 8.4.1.

After defining our metric, we must choose appropriate input datasets. Across all experiments, we rely on two textual datasets with distinct linguistic characteristics:

- *Rotten Tomatoes*: A dataset containing user-generated movie reviews from RottenTomatoes, available through HuggingFace.
- *Wikipedia*: A dataset consisting of Wikipedia articles, retrieved from HuggingFace.

For the Rotten Tomatoes dataset, see Figures 1, 2, and 3[24]. For the Wikipedia dataset, see Figure 4 and 5.

All plots are presented in **log-scale**, with color gradients representing layer depth. The use of a logarithmic scale means that approximately linear curves correspond to exponential trends in raw scale. This choice was made to emphasize the

[24] We only plot representative results for each decomposition family, based on Table 1.

early stages of training, where most of the structural convergence occurs. Interestingly, across all methods and datasets, we consistently observe two distinct regimes:

1. A **smooth convergence regime**, where the dynamics follow a near-exponential decay toward a stable configuration of concept representations.

2. A **noisy or unstable regime**, appearing later in training, where the convergence patterns fluctuate erratically depending on the decomposition method.

Interpretation: At the beginning of training, concept spaces across layers seem to converge smoothly toward a coherent representational structure. This suggests that the model initially learns a stable system of concepts that generalizes well within its activation manifold. However, as training continues, this stability gradually breaks down - leading to increasingly noisy and unstable convergence signals.

This phenomenon appears to depend strongly on the decomposition method. For instance, ICA and PCA exhibit the most pronounced instability, which aligns with the fact that both are highly sensitive to noise. In the late stages of training, when gradient magnitudes become very small, the optimizer effectively operates in a near-flat region of the loss landscape. In such regions, stochastic fluctuations dominate - effectively injecting noise into the activations. Since ICA and PCA decompose activations based on variance and independence, they are particularly susceptible to these small perturbations, which can drastically alter their decomposition trajectories.

By contrast, methods like Convex NMF tend to produce smoother late-stage behavior, likely due to their non-negativity and convexity constraints, which regularize the solution space reducing noise sensitivity.

Hypothesis: The smooth exponential regime may correspond to a phase of structured representation learning - where the model progressively disentangles meaningful features. The later noisy regime could instead reflect a kind of "representation diffusion" where fine-tuning steps lead to overfitting or concept drift. In other words, as the model approaches local minima, it stops learning stable conceptual abstractions and begins oscillating within an increasingly degenerate solution manifold. Such behavior may provide an empirical signature of overtraining or saturation in representational capacity. **But, the first regime behavior might be explained by simple model convergence...**

5.2 Lipschitz Learning Dependence

All learning dynamics show Learning Progress & Convergence (Def. 11). One could easily conclude that the representation spaces are converging, which would imply that the dynamics indicate models are indeed learning concept representations.

While this would be an intuitive interpretation, it does not fully account for the fact that extractor algorithms return specific decompositions. Here, we instead make the following hypothesis: the convergence we observe is entirely optimization-driven. More precisely, the learning process (Def. 9) is already linearly convergent[25] (at least computationally). Additionally, we will show that the extractor algorithms are Lipschitz-continuous. From there, it follows that our metrics on θ are also Lipschitz-continuous, which confirms that the convergence of the optimization process implies the convergence of the concept space.

[25] Linear convergence: given our learning process parameters, there exists $\tau > 0$ such that for all $t \in \mathbb{N}$, we have $\|\theta_{t+1} - \theta_*\| \leq \tau \|\theta_t - \theta_*\|$, where θ_* is a stationary point of our update rule.

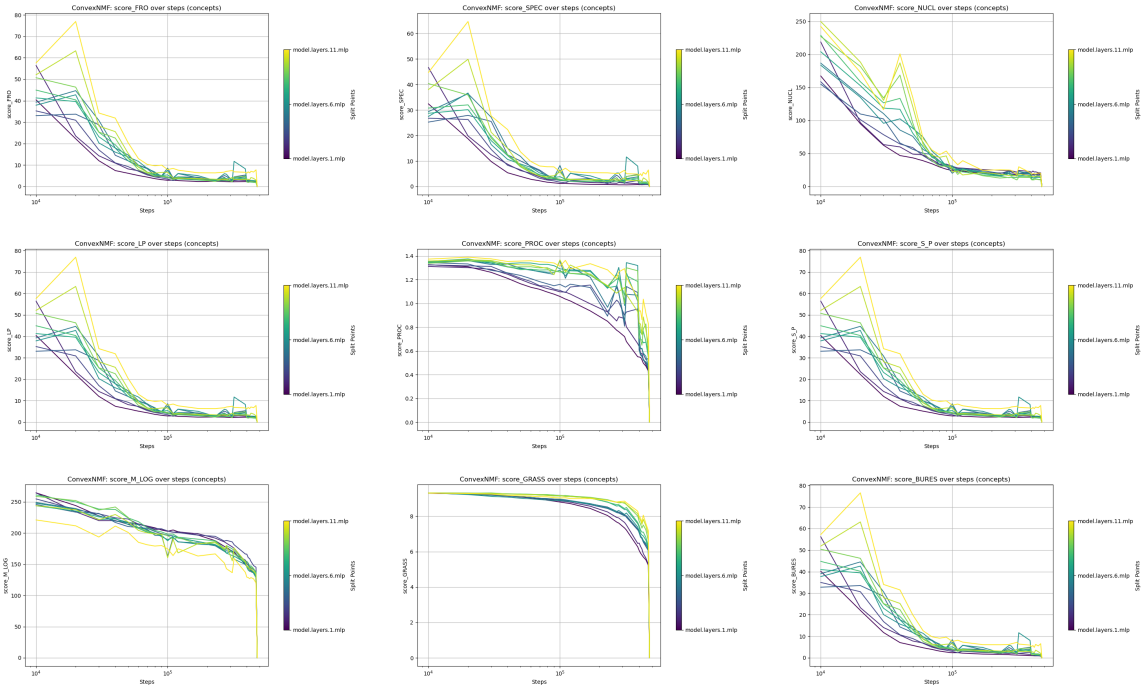

Figure 1: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be the **Convex NMF** extractor (Def. 19). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

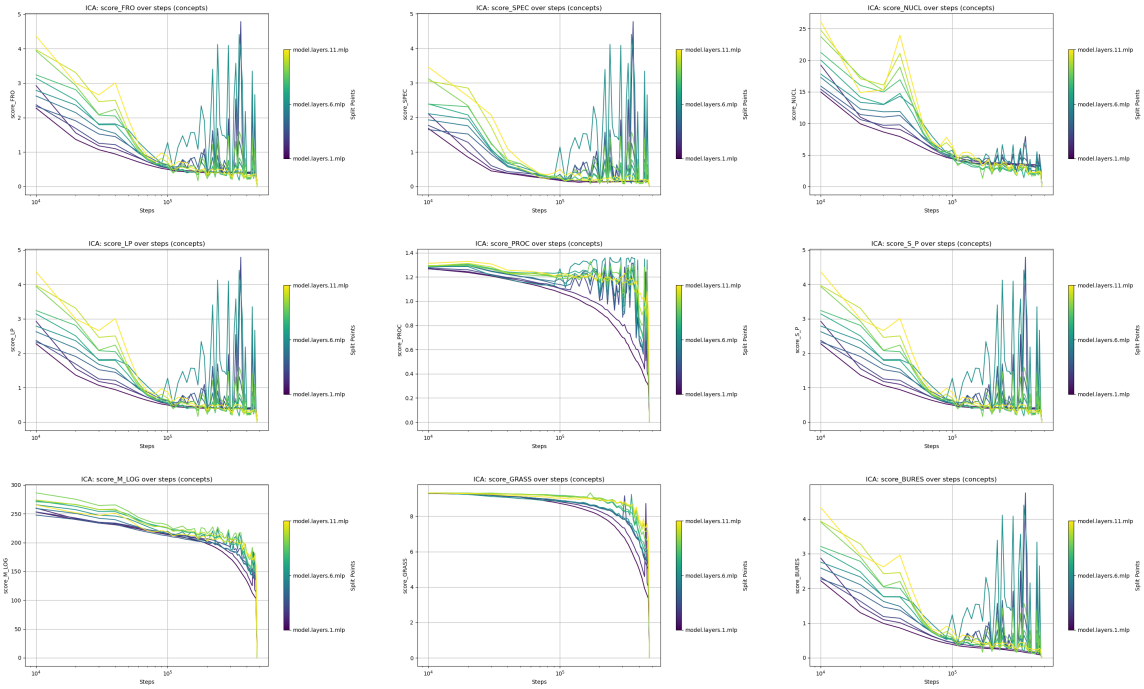

Figure 2: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be the **ICA** extractor (Def. 27). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

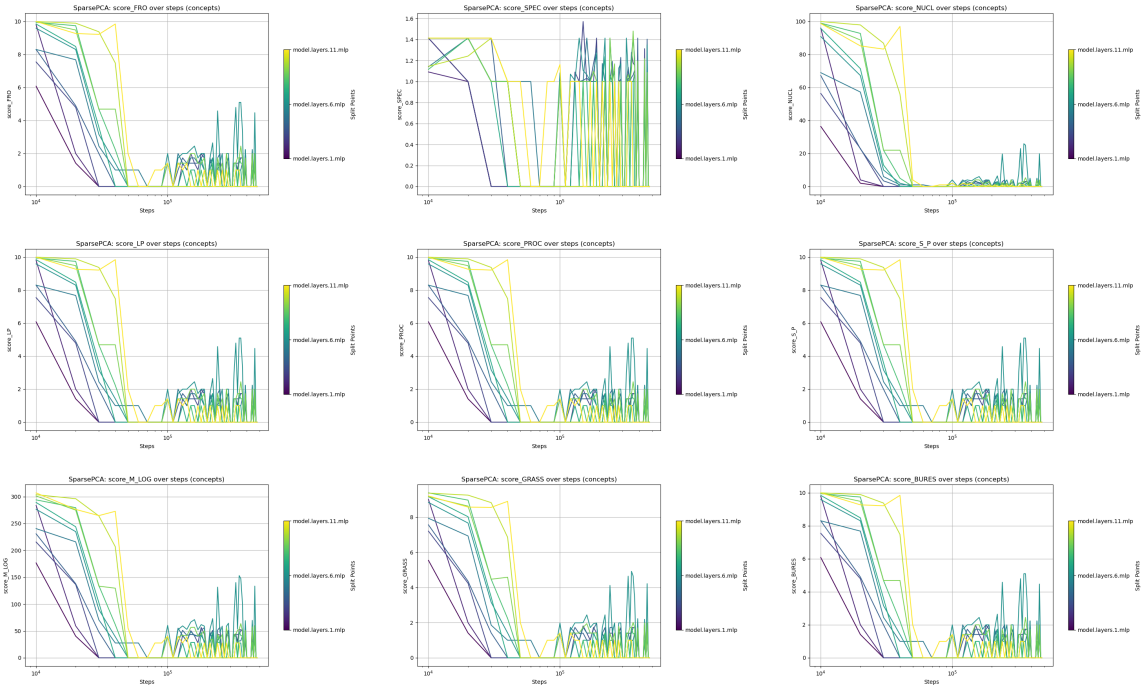

Figure 3: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be the **sparse PCA** extractor (Def. 23). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

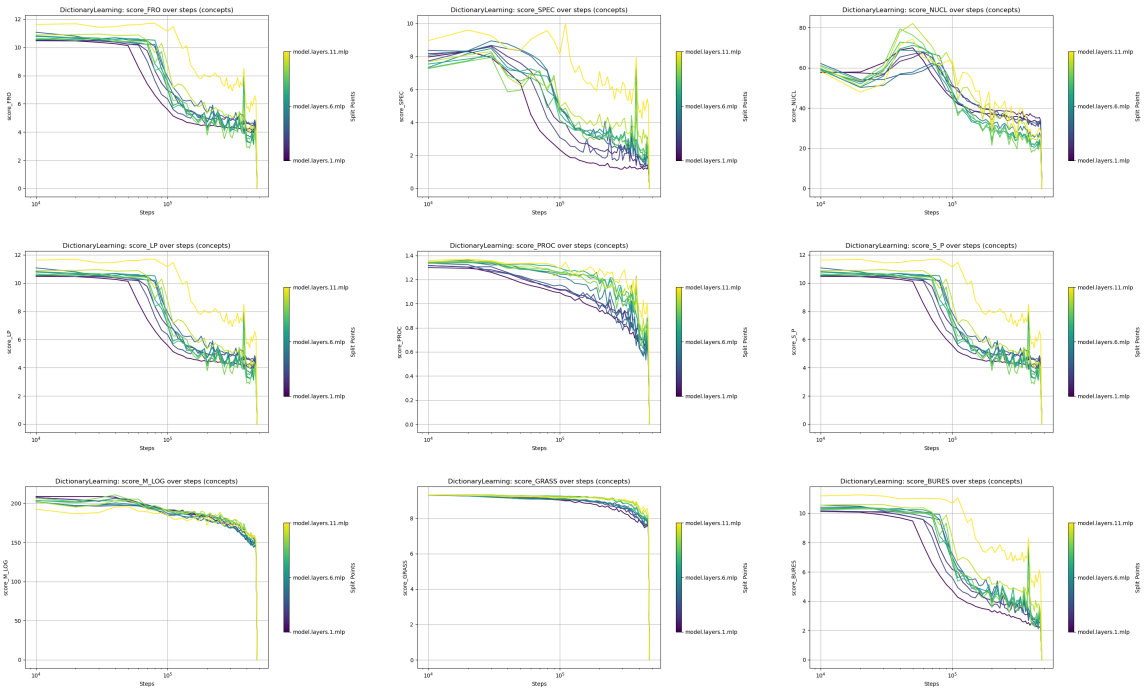

Figure 4: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be the **Dictionary Learning** extractor (Def. 29). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

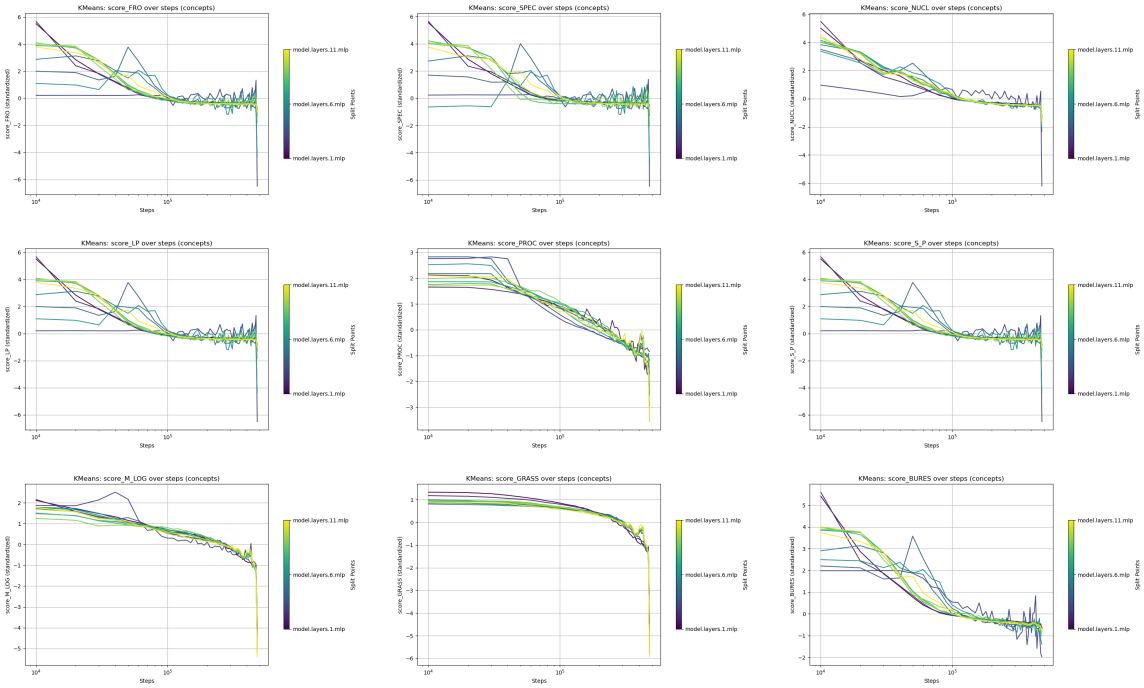

Figure 5: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be the k-**Means** extractor (Def. 28). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

Definition:

Definition 35. ***Lipschitz Continuity*** – *A function $f : \Omega \to \Gamma$ is Lipschitz continuous if and only if there exists $L > 0$ such that, for all $x, y \in \Omega$,*

$$\|f(y) - f(x)\|_\Gamma \leq L\|y - x\|_\Omega.$$

Here, $\|\cdot\|_\Omega$ and $\|\cdot\|_\Gamma$ denote norms on the spaces Ω and Γ, respectively. We will denote

$$Lip(f, \Omega) = \inf\{L \in \mathbb{R}_+ \mid \|f(x) - f(x')\|_\Gamma \leq L\|x - x'\|_\Omega,\ \forall x, x' \in \Omega\} \in \overline{\mathbb{R}}_+ = \mathbb{R}_+ \cup \{+\infty\}.$$

Let us denote $\mathscr{L} = (f_*, \mathcal{A}_L, \theta_0)$ a learning process. Let K be the depth of our model architecture and let $\{P_k\}_{k=1}^K$ be the collection of projectors, where $P_k(f_\theta, x)$ returns the activation of the k-th layer[26]. Assume we are given an unsupervised concept extractor ϕ that is Lipschitz-continuous. Furthermore, assume our distance d is norm-based, i.e., there exists a norm $\|\cdot\|$ such that $d(A, B) = \|A - B\|$. Then, using Theorem 2 (norm equivalence) and the Lipschitz continuity of ϕ, there exist $L_d > 0$ and $L_\phi > 0$ such that for all $k \in \{1, \ldots, K\}$, for all $t \in \{1, \ldots, T\}$ and all $x \in \mathcal{X}$ we have

$$d\big(\phi(P_k(f_{\theta_t}, x)),\ \phi(P_k(f_{\theta_T}, x))\big) \leq L_d L_\phi \, \|P_k(f_{\theta_t}, x) - P_k(f_{\theta_T}, x)\|.$$

Furthermore, P_k is the projector returning the k-th activation, which we will denote by $P_k(f_{\theta_t}, x) = h^k_{\theta_t}$ (according

[26] Based on the model architecture definition (Def. 6), $P_k(f_\theta, x) = P_1 \circ P_2^{(k-1)}(f_\theta(x))$ where P_1 and P_2 are simple dimension projectors on the first and second dimensions, respectively.

to the notation in Def. 6). A first result is that, under these conditions, activation convergence implies concept-space convergence with a Lipschitz dependence. Let us now examine parameter dependence:

Now assume[27] that:

- **Assumption A:** The input set $\mathcal{X}$ is bounded[28] and finite-dimensional.
- **Assumption B:** The parameter set Θ is compact[29] and finite-dimensional.
- **Assumption C:** The model f_θ is a Transformer-based LLM with activation functions $\{\sigma_i\}_{i=1}^N$ that are locally Lipschitz.

We next introduce the local Lipschitz constant. As the name implies, it is a Lipschitz constant computed locally around each point:

Definition:

Definition 36. ***Local Lipschitz Constant*** – *For a given $f : \Omega \mapsto \Gamma$, $\epsilon > 0$ and $u \in \Omega$, define*

$$Lip_\epsilon^{loc}(f, u) = \sup_{u_1, u_2 \in \mathcal{B}_\epsilon(u)} \frac{\|f(u_1) - f(u_2)\|}{\|u_1 - u_2\|}, \quad \text{where} \quad \mathcal{B}_\epsilon(u) = \{u' \in \Omega \mid \|u' - u\| \leq \epsilon\}.$$

Definition:

Definition 37. ***Locally Lipschitz Continuous*** – *A function $f : \Omega \mapsto \Gamma$ is locally Lipschitz continuous if and only if for all $u \in \Omega$ there exists $\epsilon > 0$ such that $Lip_\epsilon^{loc}(f, u) < +\infty$.*

Most neural networks are locally Lipschitz in their parameters for fixed, bounded inputs $x \in \mathcal{X}$. This is the case for Transformer-based LLMs such as EuroBERT[30]. Using **Assumptions A & C**, we can apply Property 6 and Property 8, which imply that the function $f_*(x) : \theta \mapsto f_\theta(x)$ is locally Lipschitz. Then, by Property 5 the projection onto activations is 1-Lipschitz, and the composition of locally Lipschitz functions is locally Lipschitz (Property 6). Finally, using **Assumption B** and Property 9, we deduce that the function $F_x^k : \theta \mapsto P_k(f_\theta, x)$ is globally Lipschitz over Θ. We denote $L_f(x) = Lip(f_*(x), \Omega)$ the Lipschitz constant[31].

This allows us to write:

$$d\big(\phi(P_k(f_{\theta_t}, x)),\ \phi(P_k(f_{\theta_T}, x))\big) \leq L_d L_\phi L_f(x) \|\theta_t - \theta_T\|.$$

From there we can deduce that, for a Lipschitz concept extractor, parameter convergence implies concept-space convergence. In other words, the convergence observed in our learning dynamics can be entirely explained by parameter convergence.

[27] The following assumptions are common in our setup. We are in an applied context: in typical floating-point implementations (e.g., Python), the largest finite *float* value is approximately 1.8×10^{308}. Although this is large, this implies our relevant sets are effectively bounded. Furthermore, we work in $\mathbb{R}^{|\Theta|}$, so these sets are closed and finite-dimensional, hence compact.

[28] Bounded: A set $\mathcal{X}$ is said to be bounded for a given norm $\|\cdot\|$ if and only if $\exists C > 0,\ \forall x \in \mathcal{X},\ \|x\| < C$.

[29] Compactness characterization: The set $\mathcal{K} \subset \mathbb{R}^n$ is compact if and only if for every collection of open sets $\{\mathcal{U}_i\}_{i \in I} \subset \mathcal{P}(\mathbb{R}^n)$ with $\mathcal{K} \subset \bigcup_{i \in I} \mathcal{U}_i$, there exists a finite subcollection $I_F \subset I$ such that $\mathcal{K} \subset \bigcup_{i \in I_F} \mathcal{U}_i$ (Heine–Borel).

[30] In fact, EuroBERT is not exactly a classical Transformer model. In any case, it is still locally Lipschitz: see Section 8.6.

[31] Recall that projections into coordinate dimensions are 1-Lipschitz, and the Lipschitz constant of a composition is the product of the Lipschitz constants of the composed functions. Thus, $Lip(F_x^k, \Omega) = (1)^k \times Lip(f_*(x), \Omega) = L_f(x)$.

Property:

Property 1. *Under **Assumption A, B and C**, given a lipschitz-continuous[a] concept extractor. We have that for any norm based learning dynamic, the convergence of parameters implies the convergence of said dynamic.*

[a] In fact, locally lipschitz continuous is enough.

5.3 Extractor Lipschitz Behavior

Among all extractors defined in our framework (see Section 4), the only ones that are, by definition, Lipschitz-continuous are the **SVD** and, by extension, **PCA** (as PCA is simply SVD applied to centered data). While SVD is not globally Lipschitz in general, it becomes locally Lipschitz under mild conditions-specifically, when the singular values are distinct and nonzero.[32] However, most other decomposition-based extractors (e.g., NMF, ICA, or dictionary learning methods) have no intrinsic guarantee of Lipschitz continuity in their solutions.

Why does it appear that our solutions are Lipschitz with respect to parameters? Empirically, we observe that the learned representation spaces vary smoothly with the model parameters. One might initially attribute this to chance or numerical stability; however, a deeper look reveals that this smooth dependence arises from the *algorithms* used to compute these decompositions.

Let us denote by

$$\hat{\phi} : A \mapsto (\hat{U}(A), \hat{W}(A))$$

the mapping defined by the algorithm that estimates a decomposition of A. Assume the following conditions hold:

(i) The algorithm uses only a fixed *maximum iteration* stopping criterion,[33]

(ii) Each computational step of the algorithm is locally Lipschitz with respect to its inputs,[34]

(iii) The random seed is fixed, i.e., all stochastic initializations are determined prior to execution.

Under these assumptions, the overall map $\hat{\phi}$ is a deterministic, finite composition of locally Lipschitz functions. By Property 6, such a composition is locally Lipschitz as well.

As detailed in Section 8.6, the Convex NMF algorithm satisfies these assumptions and exhibits particularly smooth convergence trajectories. Hence, among the extractors considered, Convex NMF can be viewed as one of the most Lipschitz-continuous in practice.

Finally, under **Assumptions A, B, and C**, and for any fixed input $x \in \mathcal{X}$, the activation space is compact.[35] By Property 9, local Lipschitz continuity on a compact domain implies global Lipschitz continuity on this domain. Thus, the estimated extractor $\hat{\phi}$ is globally Lipschitz on its domain.

[32] A formal argument can be derived from Corollary 3 in [YWS14]. Combining this with Weyl's inequality (see discussion on MathOverflow) establishes local Lipschitz continuity. Since our activations lie in a compact subset of $\mathbb{R}^d$, local Lipschitzness extends to global Lipschitzness on this compact domain.

[33] This implies that, even if a tolerance-based criterion exists, it is never reached before the maximum number of iterations.

[34] This holds for most common optimization algorithms, as their updates consist of compositions of smooth operations.

[35] The image of a compact set under a continuous mapping is compact. Under Assumptions A and B, fixing $x \in \mathcal{X}$ defines such a compact set. Since a locally Lipschitz function is continuous, the image of the transformer model-its activation space-is compact.

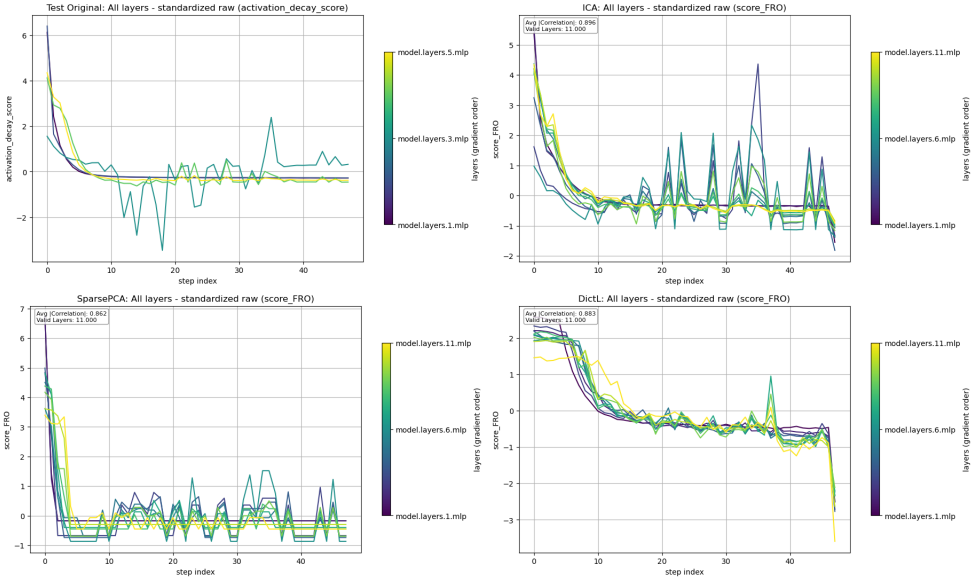

Figure 6: Standardized Frobenius **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer curves (11 Transformer layers) compared to activation dynamics. The average $|r|$ is the correlation between the rolling trend of the learning dynamics and that of the activation decay signal.

5.4 Signal Processing Comparison

Even though we have previously shown — under certain assumptions — that our learning dynamics exhibit a Lipschitz relation with both the activation and parameter spaces, it is important to emphasize that Lipschitz dependence does **not** imply identical behavior. We hypothesize that comparing the learning dynamics of a concept extractor to those of the activation space itself could offer deeper insights into how these representations evolve.

To explore this, we compute the following signal for each layer k:

$$s_t^k = \|P_k(f_{\theta_t}, x) - P_k(f_{\theta_T}, x)\|_F,$$

which we refer to as the **activation decay signal**.

Given a discrete signal $(s_t)_{t=1}^T$ and a window size $2h+1$, the *rolling mean* (or moving average) at time t is defined as:

$$\tilde{s}_t = \frac{1}{2h+1} \sum_{i=t-h}^{t+h} s_i, \tag{2}$$

where boundary indices are adjusted appropriately near the signal's edges.

We compute the rolling mean for both the activation decay signal and for the Frobenius ICA, Sparse PCA, and Dictionary Learning dynamics. We then measure their similarity using the Pearson correlation coefficient[36]. This gives us, for each Transformer layer, a measure of how closely the learning dynamic signal follows the activation decay. Finally, we average the absolute correlation values across layers (see Figure 6). We observe that the trends are highly correlated (with $|r_{xy}| > 0.8$), confirming our initial intuition: **learning dynamics are strongly dependent on activation space convergence.**

[36] Pearson correlation is defined as $r_{xy} = \frac{\sum_{i=1}^n (x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^n (x_i-\bar{x})^2}\sqrt{\sum_{i=1}^n (y_i-\bar{y})^2}}$. The closer $|r_{xy}|$ is to 1, the stronger the linear correlation between the two signals.

We then take a closer look at the relation between both curves within individual layers. In particular, ICA seems to produce a clearer separation between well-behaved and irregular layers. Layers showing irregular behavior (i.e., non-convergent activation spaces) also display irregular learning dynamics. To illustrate this, we compare two well-behaved layers (1^{st} and 11^{th}) and two irregular ones (3^{rd} and 6^{th}) - see Figure 7. For this analysis, we rely on Dynamic Time Warping (DTW)[37] and Mutual Information (MI)[38].

Interpretation: The nearly linear dependence observed between activation space convergence and learning dynamics in most layers reinforces our belief that what we are mainly observing is parameter convergence rather than concept space evolution. However, a few layers (specifically the 3^{rd} and 6^{th}) deviate from this pattern - their concept spaces behave differently from their activation spaces. Interestingly, these layers remain relatively stable during the first regime but become increasingly noisy in the second, suggesting a breakdown of local representational stability.

Hypothesis: (Disclaimer - this remains a hypothesis; deep learning processes are highly complex and difficult to interpret fully.) Recall from Section 2.1 that the capacity of large models may not stem solely from their representational complexity (see Section 8.5), but also from the behavior of the optimization algorithm itself. In particular, **Stochastic Gradient Descent (SGD)**-of which AdamW is a refined variant-exhibits two distinct phases of behavior[39]. Initially, SGD behaves like classical gradient descent, following the steepest direction of the loss landscape-this corresponds to our first, smooth convergence regime. Later, once it reaches a local minimum, the stochasticity becomes dominant, enabling the optimizer to explore flatter regions of the landscape and potentially find more general minima that satisfy multiple data batches.

This dual-phase behavior is often hidden when observing only the training loss: either the loss is too noisy to reveal the early regime or too smooth to show the later one. Our observation of the transition from smooth to noisy dynamics may thus reflect this underlying optimizer behavior. However, it remains difficult to precisely infer how these transitions relate to the **true evolution of the model's representational space**. Further analysis combining both signal and geometric representations of concept spaces could help clarify whether such noise corresponds to meaningful fine-tuning or to representational drift.

6 Going Into Representation

According to what we detailed before, a large part of the concepts convergence is due to optimization. This lead us to devise more complex methods in order to study the evolution of concepts internally.

6.1 Encoding the Representation Space

It is clear that the concept structure does not follow a linear translation through training. Indeed, it is entirely possible that, at each step, the concept directions shift slightly, meaning that observing mere convergence of concept vectors is insufficient to fully capture the underlying evolution of representations. For an intuitive illustration, see Figure 8.

To address this issue, we propose using an **encoder matrix** to quantify how similar the retrieved concept space is to the final one. The main idea is to train a dedicated encoder that learns to map activations to their corresponding concept

[37] DTW: Given two discrete signals x and y, construct a cost matrix $D \in \mathbb{R}^{(n+1)\times(m+1)}$ such that for $i, j > 0$: $D_{i,j} = |x_{i-1} - y_{j-1}| + \min(D_{i-1,j}, D_{i,j-1}, D_{i-1,j-1})$. The DTW distance is $\mathrm{DTW}(x, y) = D_{n,m}$; lower values imply more similar signals and allow for time-shifted comparisons.

[38] MI: Given discrete signals x and y, $\mathrm{MI}(x, y) \approx I(X;Y) = \sum_{u,v} p_{x,y}(u, v) \log \frac{p_{x,y}(u,v)}{p_x(u)\, p_y(v)}$. Higher MI indicates greater similarity and captures more than simple linear dependencies.

[39] For a detailed intuition, see Inference VC: Notes on Implicit Regularization in SGD.

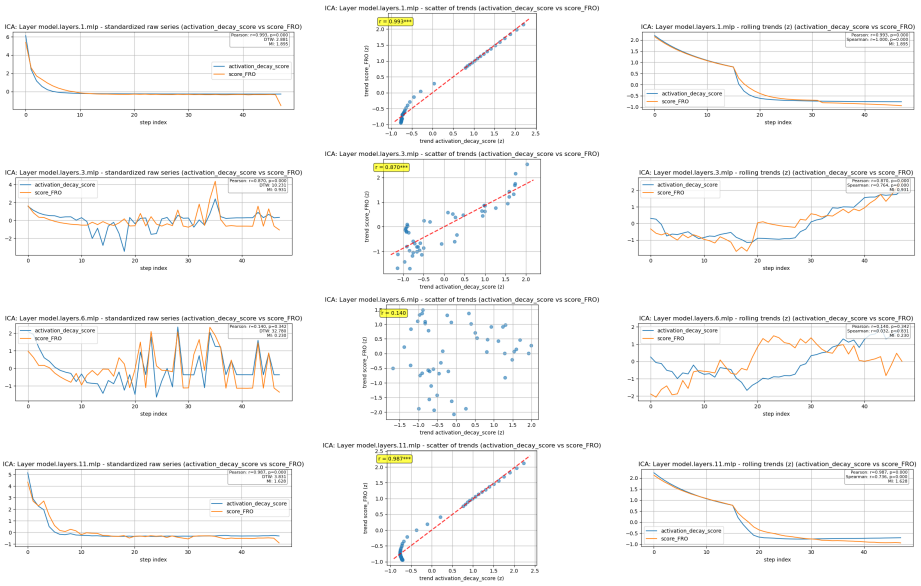

Figure 7: Time series comparison of Frobenius **Empirical Learning Dynamics** for EuroBERT-210m over 480k steps versus activation dynamics: standardized signals, scattered trends, and rolling means for the 1^{st}, 3^{rd}, 6^{th}, and 11^{th} Transformer layers. Metrics include DTW (Dynamic Time Warping), MI (Mutual Information), r (Pearson/Spearman correlation), and p (corresponding test p-values).

coefficients (also referred to as *encoded activations*). This encoder is trained on the training set at each checkpoint and then used to estimate the concept coefficients for the test set. These estimated coefficients are subsequently compared to those obtained at the final step using classical mechanistic decomposition. This process is summarized in Algorithm 3.

The intuition behind this approach is as follows: our trained (linear) encoder provides a mapping from test activations to concept coefficients, which can be interpreted as discrete distributions over the concept set. More precisely, since the coefficients are not constrained within $[0, 1]$, they are treated as *logits*[40]. Once these estimated concept distributions (encoded activations) are obtained, we compare them to the **true** concept distributions from the final model step. For this comparison, we employ classical distributional metrics such as:

- **Mean Squared Error (MSE)**[41];
- **Cross-Entropy (CE)**[42];
- **Kullback–Leibler Divergence (KL)**[43].

These metrics together provide a quantitative way to assess how the *structure* of the concept space evolves. Although this approach does not reveal how individual concepts change, it does allow us to evaluate how closely the entire space converges over training.

In Figure 9, we present the resulting learning dynamics computed using the above method. For the experiment, the *Rotten Tomatoes* dataset was split into training and testing subsets. The concept extraction and coefficient retrieval were performed using **Independent Component Analysis (ICA)**. Recall that ICA exhibited two distinct regimes in

[40] Given a discrete probability distribution $P = (p_1, \ldots, p_K)$, the logits are defined as $z_i = \log\left(\frac{p_i}{1-p_i}\right)$. To recover the original distribution from the logits, use $p_i = \dfrac{e^{z_i}}{\sum_j e^{z_j}}$.

[41] Given two discrete distributions $P = (p_1, \ldots, p_K)$ and $Q = (q_1, \ldots, q_K)$, $\mathrm{MSE}(P, Q) = \frac{1}{K}\sum_{i=1}^{K}(p_i - q_i)^2$.

[42] Given $P = (p_1, \ldots, p_K)$ and $Q = (q_1, \ldots, q_K)$, $\mathrm{H}(P, Q) = -\sum_{i=1}^{K} p_i \log(q_i)$.

[43] Given $P = (p_1, \ldots, p_K)$ and $Q = (q_1, \ldots, q_K)$, $\mathrm{KL}(P \,\|\, Q) = \sum_{i=1}^{K} p_i \log\left(\frac{p_i}{q_i}\right)$.

previous analyses of concept vector comparison: a first *smooth exponential* regime (from step 0 to approximately 100,000), and a second *noisy* regime. To emphasize this latter phase, we do not use a logarithmic scale for the plots, although the early regime still reflects an exponential decay.

Interestingly, in Figure 9, the curves appear substantially smoother overall (except for early instabilities likely due to correlation noise). A particularly striking observation is that the previously noisy second regime of ICA now appears much more regular and stable when examined through the lens of this encoding-based comparison.

Interpretation: This smoother behavior suggests that much of the apparent "noise" in direct concept-to-concept comparisons may stem from minor reorientations of the concept basis rather than genuine instability in the underlying representation structure. By aligning representations through a trained encoder, we effectively factor out these small geometric shifts, revealing a more coherent and continuous progression of the concept space. This supports the idea that the high-level representation dynamics of large models may be more stable-and potentially more structured-than they appear under naive vector-space comparisons.

Algorithm 3: Activation-based Representation Learning Algorithm

input : Number of steps T;
Training activation matrices $\{A_{tr}^t\}_{t=1}^T$;
Test activation matrices $\{A_{te}^t\}_{t=1}^T$;
Norm to use for the reconstruction error $\|\cdot\|$;
Concept extractor ϕ.

output: Encoded activations $\{U_{te}^t\}_{t=1}^T$.

Compute last activation decomposition for training:
$(U_{tr}^T, W_{tr}^T) \leftarrow \phi(A_{tr}^T)$
Compute last activation decomposition for test:
$(U_{te}^T, W_{te}^T) \leftarrow \phi(A_{te}^T)$
for *each* $t = 1$ **to** $T - 1$ **do**
Find encoder E_{tr}^t minimizing the reconstruction error:

$$E_{tr}^t \leftarrow \arg\min_E \|U_{tr}^T - A_{tr}^t E\|$$

Compute test representation:
$U_{te}^t \leftarrow A_{te}^t E_{tr}^t$

Hypothesis: By training an encoder E_t to estimate U_t on test data, we have effectively broken the Lipschitz dependence between the activation space and the concept space. Nonetheless, we still observe linear convergence of the concept space. However, this convergence now reflects the **structure** of the concept space, rather than the convergence of the raw space itself. The resulting dynamics appear smoother overall, suggesting that the concept space structure may have already converged during the first regime, while the model's stochastic evolution in the second regime is primarily driven by residual instability in the concept vectors. In other words, the model first learns to identify and disentangle the core concepts (during the early regime) and subsequently refines and stabilizes them during the stochastic phase of training, leading to the emergence of more robust conceptual representations.

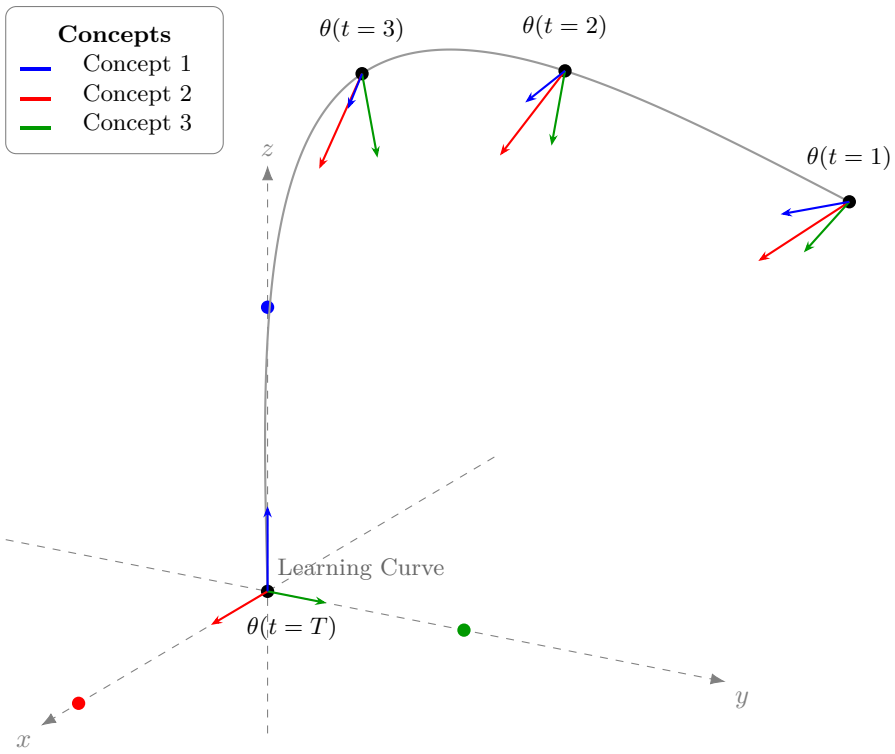

Figure 8: Evolution of θ over discrete time steps, with vectors indicating orientations toward three fixed concepts. The vector lengths represent the coefficients associated with each concept. The main intuition behind this figure is that, during training, the model may learn similar concepts but represent them differently (here, concept directions do not evolve linearly over time). In such a case, our process may fail to properly track concepts for direct comparison. However, the relative importance of concepts should remain stable-hence, we should focus on comparing their significance rather than their raw orientation (illustrated here by the arrow lengths).

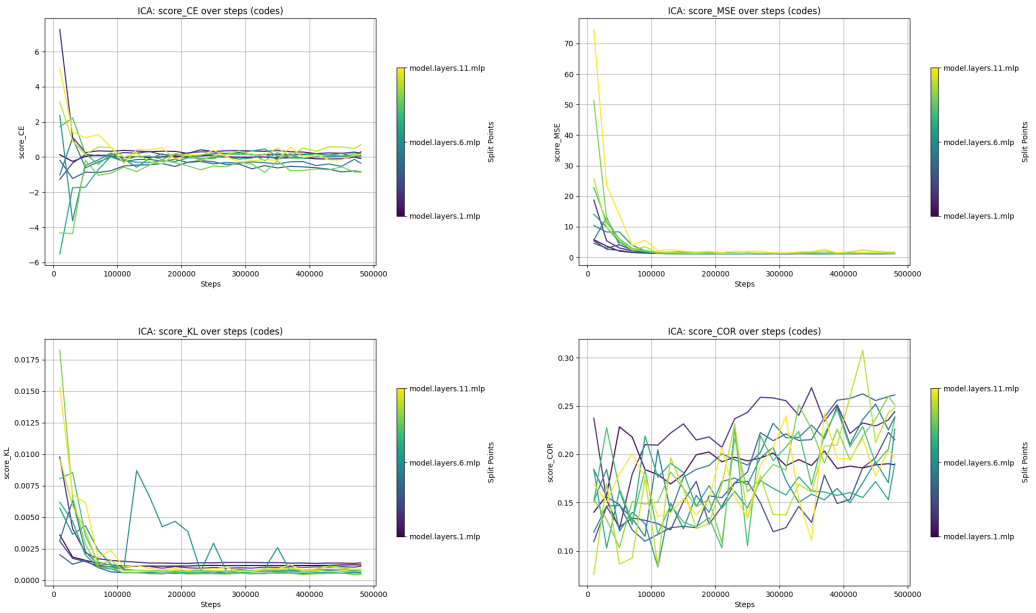

Figure 9: Encoder-based concepts distribution **Empirical Learning Dynamics** for EuroBERT-210m over 480k steps. Distribution comparison metric (left→right): CE (Cross-Entropy), MSE (Mean Squared Error), KL (Kullback-Leibler divergence) and COR (Correlation between logits).

6.2 Giving Meaning to Concepts: CAV

Concept Activation Vectors (CAV) were introduced by Kim et al. [KWG+18] as a way to link a model's internal representation to human-interpretable concepts. Unlike unsupervised decomposition methods (such as NMF, ICA, or Dictionary Learning), CAV relies on labeled examples of a target concept. In our framework, this makes it a **supervised concept extractor**.

Definition:

Definition 38. ***CAV Extractor (supervised)*** *- Given a decomposable model $f = g \circ h$, let $\mathcal{D}_c^+ \subset \mathcal{X}$ be a dataset of positive examples representing a target concept c, and $\mathcal{D}_r^- \subset \mathcal{X}$ a set of negative examples. Assume[a] we have a function $\mathcal{G} : \mathcal{X} \longrightarrow \{0, 1\}$ such that $\mathcal{G}(\mathcal{D}_c^+) = \{1\}$ and $\mathcal{G}(\mathcal{D}_c^+) = \{0\}$. We define the CAV extractor (See Def 13) as the mapping*

$$\Phi\big(f_\theta,\ \mathcal{D}_c \cup \mathcal{D}_r\big) \ = \arg\min_v \{\mathbb{E}_{x \sim \mathcal{D}_c \cup \mathcal{D}_r}[\ell_{CE}\big(h_\theta(x).v, \mathcal{G}(x)\big)]\},$$

Where the loss ℓ_{CE} is the cross-entropy loss but could be any classical loss. Assuming we are computing CAV for k^{th} layer, we will note it v_c^k.

[a] This is "Homo In Machina", a function representing human labeling datasets (Supervised setting).

This simple construction allows us to associate a semantic/ syntactic concept (defined through examples) with a direction in the activation space.

Testing with CAV (TCAV) extends this idea to evaluate the *influence* of a concept on the model's prediction. Given a target class y, TCAV estimates how sensitive the model output $f_{\theta,y}(x)$ is to perturbations along the concept direction v_c^k. This is achieved by computing the directional derivative of $f_{\theta,y}$ with respect to the activations $P_k(f_\theta, x)$ along v_c^k.

Formally, the *TCAV score* is defined as:

$$\text{TCAV}_{y,c}^k = \mathbb{E}_{x \sim \mathcal{D}_y}\left[\mathbb{I}\left(\frac{\partial f_{\theta,y}(x)}{\partial P_k(f_\theta, x)} \cdot v_c^k > 0\right)\right],$$

where $\mathbb{I}(\cdot)$ denotes the indicator function. It measures the fraction of inputs for which moving in the direction of the concept increases the probability of predicting class y.

Intuition: CAV identifies a concept direction in the activation space by contrasting concept positive examples against negative one[44], while TCAV measures how much this direction influences the model's prediction. Unlike unsupervised extractors, which reconstruct activations as combinations of latent concepts, CAV/TCAV directly link the activation geometry to semantically meaningful features defined by the user.

The overall pipeline remains identical: for every training step and layer, we extract the corresponding CAVs while simultaneously computing the TCAV scores. In our setup, we investigate both *semantic* and *syntactic* concepts. To achieve this, we rely on subsets of the GLUE benchmark dataset.

GLUE includes a subset named `sst2`, which consists of short movie reviews labeled as positive or negative (1/0). This subset naturally represents our **semantic** concepts—the distinction between "good" and "bad". Another subset, `cola`, contains sentences labeled according to grammatical correctness (1/0), and thus provides our **syntactic** concepts. From

[44] Here, positive and negative can means different things : For example it can be the presence/ absence of a concept but it can also be a dual concept with positive and negative examples (The concept of compliments/ critics).

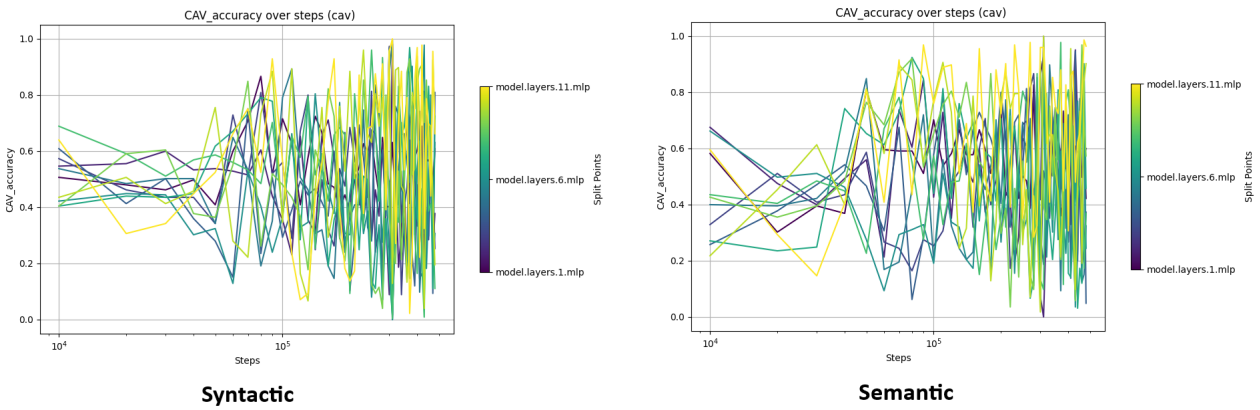

Figure 10: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). We use the CAV extractor (Def. 38). Each dynamic uses distance $m_t = TCAV_t$.

each subset, we sample 2,000 sentences, using 2/3 of them to train the CAVs, and keeping the remaining third to compute TCAV.

As EuroBERT is a model trained on the Masked Language Modeling (MLM) task, we evaluate TCAV with respect to the token prediction class y. Figure 10 illustrates the resulting dynamics: once again, two regimes appear. The first is relatively smooth, followed by a second that is significantly noisier (Note that the use of log-scale in the plot can make this noise appear more compact than it truly is). To provide a more systematic overview, we summarize the regime-wise means and standard deviations of TCAV in Table 2. To ensure regime comparability, the first regime is measured between steps 20,000 and 100,000, while the second regime spans from 380,000 to 460,000.

	Syntactic			Semantic		
Layer	**First**	**Second**	**Overall**	**First**	**Second**	**Overall**
1	**0.597 (0.145)**	0.440 (0.115)	0.514 (0.124)	0.571 (0.137)	**0.581 (0.109)**	0.561 (0.142)
2	0.514 (0.077)	**0.594 (0.184)**	0.498 (0.184)	0.535 (0.186)	**0.635 (0.179)**	0.536 (0.195)
3	**0.543 (0.146)**	0.480 (0.150)	0.473 (0.202)	0.329 (0.128)	**0.339 (0.191)**	0.455 (0.201)
4	**0.385 (0.180)**	0.341 (0.237)	0.445 (0.220)	0.423 (0.183)	**0.577 (0.294)**	0.514 (0.220)
5	0.483 (0.176)	**0.506 (0.264)**	0.496 (0.224)	**0.441 (0.235)**	0.367 (0.201)	0.441 (0.205)
6	0.342 (0.116)	**0.574 (0.250)**	0.443 (0.221)	0.308 (0.126)	**0.528 (0.270)**	0.427 (0.219)
7	**0.606 (0.156)**	0.484 (0.221)	0.479 (0.240)	**0.650 (0.205)**	0.277 (0.232)	0.463 (0.252)
8	**0.541 (0.087)**	0.337 (0.203)	0.504 (0.228)	0.502 (0.175)	**0.541 (0.248)**	0.473 (0.232)
9	0.501 (0.168)	**0.725 (0.162)**	0.514 (0.251)	**0.663 (0.173)**	0.572 (0.291)	0.541 (0.241)
10	0.512 (0.230)	**0.619 (0.247)**	0.566 (0.258)	**0.684 (0.162)**	0.491 (0.284)	0.548 (0.263)
11	0.593 (0.172)	**0.657 (0.275)**	0.573 (0.284)	0.606 (0.277)	**0.678 (0.190)**	0.672 (0.241)
$\Sigma\Delta$		**+0.142**			**-0.127**	
$\Sigma\omega\Delta$		**+0.125**			**-0.292**	

Table 2: TCAV scores (mean and SD) per layer and regime. Values rounded to three decimals; parentheses show SD. For each layer and modality, the larger of the **First** / **Second** regime means is shaded (gray) and typeset in bold. Between the two **Overall** columns (Semantic vs Syntactic) the lower overall mean is colored blue and the higher is colored red. The row "**$\Sigma\Delta$**" reports the simple sum over layers of (Second − First) regime means (semantic: +0.142; syntactic: −0.127). The row "**$\Sigma\omega\Delta$**" is a σ-aware aggregate: it sums per-layer standardized changes $\Delta_i/\sqrt{\sigma_{1,i}^2 + \sigma_{2,i}^2}$ (with Δ_i the per-layer mean difference and $\sigma_{1,i}, \sigma_{2,i}$ the corresponding SDs). Positive values indicate a net increase that is large relative to typical per-layer variability.

Interpretation: Our first observation is that **syntactic concepts** tend to benefit more from the second training regime

overall. Nevertheless, semantic concepts show improvement across 7 of the 11 layers, while syntactic ones improve in only 6 layers. This suggests that while the second regime enhances some layers' conceptual sensitivity, it does not guarantee uniform improvement across the network.

We propose two complementary explanations for this phenomenon:

- **Increased noise:** Most of the layers showing an increase in TCAV also display a larger standard deviation. This supports the idea that the second regime is inherently more stochastic, possibly due to the optimizer's implicit regularization effects.
- **Concept diffusion:** During the second regime, concept representation appears to spread more evenly across layers. For semantic concepts, this diffusion slightly reduces overall accuracy but increases the number of layers capable of identifying and propagating the concept toward the deeper parts of the model. Notably, the last layer—typically associated with higher-level abstraction—shows both stronger and more stable scores. This aligns with common findings in mechanistic interpretability, where deeper layers are known to encode more complex or abstract concepts.

6.3 The Concepts Manifold

So far, we have observed two regimes that exhibit different behaviours in the model's learning dynamics. However, we have not yet examined the concepts themselves. After performing PCA and inspecting matrix transformations of the concept space (see Section 8.7), it became apparent that the concepts - considered as data (we have r concept vectors of dimension m) - lie on a complex manifold.

We give an intuitive definition of a manifold for our setting:

Definition:

Definition 39. ***Manifold (Intuitive Definition)*** — *A subset $M \subset \mathbb{R}^m$ is called a* d-dimensional manifold *if, for every point $p \in M$, there exists a neighbourhood $U \subset \mathbb{R}^m$ and a C^1 function $f : U \to \mathbb{R}^{m-d}$ such that*

$$M \cap U \;=\; \{\, x \in U \mid f(x) = 0 \,\},$$

and the Jacobian matrix $\mathcal{J}_f(x)$ has full rank $m - d$ for all $x \in M \cap U$.
Intuitively, M is a "smooth surface" in $\mathbb{R}^m$ that locally looks like $\mathbb{R}^d$, much as a curve ($d = 1$) or a surface ($d = 2$) does in three-dimensional space.

A central question in manifold analysis is to determine the dimension d, also called the *intrinsic dimension*. Many methods exist to estimate this quantity from a dataset (assuming it is sampled from a manifold possibly corrupted by small noise): some estimators are global, others are local. In our setting we prefer local methods because the behaviour of our solutions is not well characterised a priori. A simple, robust and easy-to-implement local estimator is **TWO-NN** [FdRL17].
Mathematical background (TWO-NN = 2 Nearest Neighbours): Let $W = \{w_i\}_{i=1}^n \subset \mathbb{R}^m$ be a point cloud (here rows are averaged concept vectors). Denote by $r_1(w_i)$ and $r_2(w_i)$ the Euclidean distances from w_i to its first and second nearest neighbours (excluding itself). Define the ratio

$$\mu_i := \frac{r_2(w_i)}{r_1(w_i)} \qquad (i = 1, \ldots, n).$$

Algorithm 4: TWO-NN on averaged concept matrices (per layer and per regime) [FdRL17].

input : $x = \{x_j\}_{j=1}^{N}$: test dataset (batch size $B = N$);
f_θ: model architecture (decomposable model / mapping $(x, \theta) \mapsto h_\theta(x)$);
$\{\theta_t\}_{t=0}^{T}$: snapshot parameters at timesteps $0, \dots, T$;
$\{P_k\}_{k=1}^{K}$: list of projectors where $P_k(f_\theta, x)$ returns activations of the k-th layer;
ϕ: concept extractor, $\phi(h) \mapsto (U, W)$ with $W \in \mathbb{R}^{m \times r}$;
$T^{(1)}, T^{(2)} \subset \{0, \dots, T\}$: index sets for first and second regimes.

output: Intrinsic dimension estimates $\hat{d}_k^{(i)}$ for each layer k and regime index $i \in \{1, 2\}$.

Initialize arrays $\hat{d}^{(1)}[1:K] \leftarrow 0$, $\hat{d}^{(2)}[1:K] \leftarrow 0$

for *each layer* $k = 1$ **to** K **do**

  for *each checkpoint* $t = 0$ **to** T **do**

    Compute activations for all samples: $H_t^{B,k} \leftarrow \big(P_k(f_{\theta_t}, x_j)\big)_{j=1}^{N} \in \mathbb{R}^{N \times m}$

    Compute mean activation: $\bar{H}_t^k \leftarrow \frac{1}{N} \sum_{j=1}^{N} H_{t,j}^{(k)} \in \mathbb{R}^m$

    Extract concepts from mean activation: $(U_t^k, W_t^k) \leftarrow \phi\big(\bar{H}_t^k\big)$, with $W_t^k \in \mathbb{R}^{m \times r}$

  for *each regime index* $i \in \{1, 2\}$ **do**

    Compute averaged concept matrix over the regime:

$$\overline{W}^{k,i} = \frac{1}{|T^{(i)}|} \sum_{t \in T^{(i)}} W_t^k \quad \in \mathbb{R}^{m \times r}.$$

    Form the point cloud for TWO-NN as the rows of $(\overline{W}^{k,i})^\top \in \mathbb{R}^{r \times m}$ (each row is a concept vector in $\mathbb{R}^m$)

    for *each concept row* $j = 1$ **to** r **do**

      Compute $r_{1,j}$ = Euclidean distance from row j to its nearest neighbour among the r rows

      Compute $r_{2,j}$ = Euclidean distance from row j to its second nearest neighbour

      Compute ratio $\mu_j \leftarrow r_{2,j}/r_{1,j}$

    Sort $\{\mu_j\}_{j=1}^{r}$ ascending to obtain $\mu_{(1)} \le \cdots \le \mu_{(r)}$

    Compute empirical values $S_j \leftarrow 1 - j/r$ for $j = 1, \dots, r$

    Set $x_j \leftarrow \log \mu_j$ and $y_j \leftarrow \log S_{(j)}$ for $j = 1, \dots, r$

    Compute linear regression slope b of y on x (ordinary least squares)

    Set $\hat{d}^{(i)}[k] \leftarrow -b$

return $\{\hat{d}^{(1)}[k], \hat{d}^{(2)}[k]\}_{k=1}^{K}$

Under the assumption that, locally, the points density is uniform[45] in an unknown intrinsic dimension d, one can show that

$$P(\mu \le t) = (1 - t^{-d})\mathbb{I}_{t \ge 1},$$

Equivalently, the cumulative distribution function of μ satisfies $\log(1 - P(\mu \le t)) = -d \log t$. TWO-NN exploits this relation to estimate d by fitting a linear model to $\log(1 - P(\mu \le t))$ versus $\log t$.

Our plan is the following. We run Algorithm 4 over the two regimes defined for training. Concretely, the first regime will cover the early phase (from 10,000 to 240,000 steps) and the second regime the later phase (from 250,000 to 480,000 steps). We choose these wide intervals to reduce sampling noise: noisy data tend to inflate intrinsic-dimension estimates, whereas averaging over many checkpoints lowers embedding dimension and yields more robust estimates (see the discussion of the curse of dimensionality for kNN in [Cornell Lecture Note]).

[45] This methods robustness comes from the fact that this hypothesis only need to be verified for the two nearest neighbors, which is often the case.

Intuition: By averaging $\phi(P_k(f_{\theta_t}, x))$ over checkpoints inside each regime we obtain concept representations that summarize the model's behavior in that phase of training. Treating those averaged vectors as a point cloud captures the *typical* representation geometry for the regime (rather than transient fluctuations at single checkpoints). TWO-NN then provides a lightweight, local statistic of how many degrees of freedom are effectively used by the concept vectors: a small $\hat{d}$ suggests the averaged concept vectors lie near a low-dimensional manifold (strong structure / redundancy), while a large $\hat{d}$ indicates higher intrinsic complexity.

Comparing $\hat{d}_k^{(1)}$ and $\hat{d}_k^{(2)}$ for each layer k therefore quantifies how the *intrinsic complexity* of the concept representation changes between the early (smooth) and late (noisy / stochastic) regimes of training. We show results[46] in Table 3 : We make a separation between sparse & non-sparse extractors since intrinsic dimension under-perform when faced with sparse data.

Layer	Non-sparse methods						Sparse methods			
	SVD		ICA		Semi-NMF		Dict. Learning		K-Means	
	First	Second	First	Second	First	Second	First	Second	First	Second
1	34.91	24.70	46.98	31.09	34.06	24.95	13.17	15.25	11.10	15.57
2	28.72	25.05	37.94	23.79	27.04	27.97	12.35	13.58	9.61	9.98
3	35.84	34.02	23.14	8.55	22.77	11.29	9.76	10.19	3.05	2.52
4	40.35	34.27	17.54	16.51	20.01	12.73	11.20	8.03	3.54	4.01
5	42.18	37.83	22.85	20.86	21.58	18.38	8.67	7.83	3.39	3.91
6	46.52	36.01	14.58	9.99	10.42	8.76	7.28	7.79	3.18	3.13
7	36.96	37.08	29.10	21.27	15.24	24.22	7.33	6.00	6.32	6.30
8	49.08	38.95	20.17	21.99	40.13	22.32	9.04	6.04	8.58	8.67
9	48.40	55.21	20.69	14.67	20.96	13.78	10.69	8.21	8.45	17.47
10	75.07	57.50	44.47	37.67	34.07	32.38	10.64	7.12	7.74	14.02
11	68.98	55.10	20.36	20.79	24.42	23.50	8.30	5.69	4.78	6.70
Mean	46.09	39.61	27.07	20.65	24.61	20.03	9.86	8.70	6.34	8.39

Table 3: Estimated intrinsic dimension per layer and regime. For each method we report the block-mean intrinsic dimension in the **First** and **Second** regimes. The **gray** cell indicates which regime (first or second) yields the *lower* intrinsic dimension for that layer-method pair. The final row (**Mean**) gives the average intrinsic dimension across layers.

Interpretation: Across nearly all extraction methods, we observe a clear *reduction* in intrinsic dimension during the second regime (in means). This suggests that the model's learned concepts become more structurally dependent-indicating a more compact and coherent concept organization. The only notable exception is k-Means, for which intrinsic dimension estimators are known to be less reliable because of the sparse settings. Moreover, in sparse extraction methods, random noise tends to dominate over the continuous structural variations, which can obscure fine-grained dependencies. It is crucial to understand that a reduction in intrinsic dimension generally reflects *improved generalization.* Conversely, an overfitted model would exhibit a maximized intrinsic dimension-where each concept corresponds almost directly to individual data points.

Hypothesis: In continuity with our reasoning that the noisy second regime contributes to generalization, these results are largely consistent with our expectations. The intrinsic dimension of the concept manifold decreases during the second phase of training. *Why was this predictable?* As previously discussed, the existence of two regimes implies distinct roles: the second regime allows the algorithm's inherent stochasticity to explore and refine local minima. During this stage, the model is exposed to different data batches at each iteration, introducing substantial noise in the activations- *yet not in the concept structure* (as shown in Section 6.1). We hypothesize that throughout this regime, the model fine-tunes its internal

[46]Note that we report results for *semi-NMF* rather than *convex NMF*. This choice stems from the fact that nearest-neighbor algorithms cannot operate on data containing missing values (`NaN`). Unfortunately, convex NMF occasionally produces such `NaN` values during computation.

representation of concepts so as to generalize across the varying inputs it encounters. Under this perspective, we can infer that the model's strong generalization ability stems from this dual-phase dynamic-where the second, stochastic regime plays a central role in refining the conceptual structure.

7 Conclusion

Throughout this work, we aimed to study the learning dynamics of a large language model. To achieve this, we employed mechanistic interpretability (MI) methods specifically designed to peer inside large models. While many of these methods are not originally from MI, we proposed a general MI framework to unify and clarify this part of the domain.

From this framework, we constructed empirical learning dynamics. By showing that most classical MI methods exhibit strong (Lipschitz) parameter dependence, we observed that their convergence behavior could often be explained by the parameter dynamics themselves. Nonetheless, these methods allowed us to efficiently highlight the presence of two distinct training regimes. Although the existence of two training phases in LLMs was previously known, it remained difficult to observe directly from the training loss, which is often either too noisy or too smooth.

Then, we removed the Lipschitz dependence by focusing on the concept distribution rather than the individual concepts. We observed that, while concept vectors are relatively noisy during the second regime, their overall distribution remains stable. Although the function applied to the encoded activation may contribute to this effect, it does not fully explain the smoother and more convergent behavior we observed-thus supporting our hypothesis of preserved concept distributions.

Subsequently, using TCAV, we sought to give semantic meaning to the activation concepts studied. Since CAV is a supervised extractor, it enables such interpretability. The results revealed that both regimes are also reflected in human-level concepts, even if the interpretation remains challenging.

Finally, motivated by the intuition behind the existence of these two regimes and after further investigating concept structures, we concluded that studying the underlying manifold on which concepts evolve was necessary. By assessing the intrinsic dimension of the concept manifold for both regimes, we made a significant observation: the intrinsic dimension decreases from the first to the second regime. This strongly supports the hypothesis that the stochasticity of the second regime serves as a concept refinement process aimed at improving generalization-an intuition reinforced by the observation that stochasticity itself appears to drive the model in that direction.

In conclusion, while we have confirmed the existence of a two-phase training process in large language models, we have also gained valuable insight into why larger models tend to generalize better when trained longer. This opens up many promising directions for future work, such as developing more robust estimators for the intrinsic dimension in the context of extracted concepts. One could, for instance, design *decomposer-aware* intrinsic dimension estimators, since most extractors yield structured representations. Other potential directions include a deeper analysis of differences between models within the EuroBERT family (see Section 8.8.1), or exploring further the link between model convergence and concept convergence using novel approaches, such as the encoder-based method introduced here.

A final avenue for exploration would be to extend this framework to multimodal or continual learning settings, assessing whether similar dual-regime dynamics and concept refinements emerge when models learn across multiple domains or over time.

8 Annex

8.1 Glossary

Stochastic Stochastic refers to the randomness of a system. It is stochastic if it has a non-determinate part.

Token "Every sentence is made of multiple letters." This vacuous sentence contains about 42 characters. If we were to feed a language model with individual letters, it would quickly become inefficient. To address this, researchers have developed statistical methods that group letters together based on how frequently they appear next to each other. These grouped units are called **tokens** (typically, one token corresponds to around 5 characters).

Dataset A **dataset** is a collection of samples drawn from a specific distribution that we aim to learn from using machine learning methods.

Batch A batch is a subset of the dataset for which the model will be trained on at one step. Choosing randomly the batch allows for stochastic learning.

Pre-trained A **pre-trained model** is a model that has been trained on a general-purpose task, usually over a large dataset, before being adapted to other tasks.

Post-hoc A study is said to be **post-hoc** when it is conducted on a pre-trained model without modifying its parameters.

Fine-tuning **Fine-tuning** refers to the process of retraining a pre-trained model on a new, typically smaller, task-specific dataset to adapt it to a new objective.

Syntactic In natural language, something is said to be **syntactic** when it relates to the rules and structure of the language (i.e., grammar).

Semantic In natural language, **semantic** refers to the meaning or sense that words and sentences convey-it is the field of abstract conceptual interpretation.

Transformer The **Transformer** (see Definition 5) is the most widely used architecture in large language models. It relies on an *attention mechanism* that allows the model to process data while weighting the importance of different elements within the input sequence.

BERT **BERT** (Bidirectional Encoder Representations from Transformers) is a model introduced by Google in 2018 that demonstrated how transformer-based architectures could dramatically improve performance in natural language processing tasks. Today, *BERT* refers to a family of models built upon transformer encoder structures.

Disentanglement **Concept disentanglement** can be defined in multiple ways. Here, we use it to describe the phenomenon by which a model successfully separates distinct concepts within its internal representations.

Embedding An **embedding** is a high-dimensional vector obtained by applying a mathematical function that maps elements from a lower-dimensional space to a higher-dimensional one.

Manifold A **manifold** is a mathematical object describing a geometrically coherent structure embedded in a higher-dimensional space. For example, a donut (torus) is a 2-dimensional manifold embedded in 3 dimensions-one can move in only two directions on its surface.

Gradient Often denoted by a nabla symbol (∇), representing the direction in which a function has the steepest ascent.

Lipschitz constant A constant that bounds how much a function's value can change with respect to changes in its input. The smaller this positive constant, the flatter the function.

Adam An optimization algorithm that combines the advantages of two other extensions of stochastic gradient descent (SGD).

SGD Stochastic Gradient Descent, an optimization method that updates parameters iteratively based on random subsets of data.

Local & Global Optimum A point on which a function is maximal (minimal) locally or globally. A local maximum (minimum) is a maxima on a subset around our point.

Bounded A normed set is said to be bounded if for every element in the set, its norm is under a value $M > 0$ independent from this element.

Compact A normed set is said to be compact in finite dimension if it is closed and bounded (In infinite dimension, it must verify Heine-Borel condition).

8.2 Parameters & Code

The code can be found here [GitHub]. It often relies on the library interpreto upon which I worked a bit, I strongly recommend taking a look : [InterpretoGit]. Unfortunately, interpreto library may evolves in the futur which implies that the compatibility with the code may be lost.

For the parameters general settings we have (if not mentioned otherwise):

- **Number of concepts** $r = 100$.
- **Number of samples** We used approximately 5,000 tokens (around 20,000 `strings`).
- **CAV learning** Learning rate $\eta = 0.01$ for 2,500 iterations at each training step.
- **CAV regularization** Norm-based regularization with coefficient $\lambda = 0.3$.
- **Other parameters** For all remaining hyperparameters, we used previously selected values (sometimes with a pre-implemented scheduler).

8.3 Annex A: Notations & Properties

Notations :

Notation 1. $\mathbb{R}^{n\times m}$ *: Set of real-valued matrices with dimension* n *times* m*.*

Notation 2. $\mathbb{R}_+^{n\times m}$ *: Subset of* $\mathbb{R}^{n\times m}$ *with non-negative coefficient.*

Notation 3. $\subset$ *&* $\subsetneq$ *:*

$\subset$ *: We note* $A \subset B$ *if and only if* $\forall a \in A,\ a \in B$ *i.e.* $A \cup B = B$*.*

$\subsetneq$ *: We note* $A \subsetneq B$ *if and only if* $A \subset B$ *and* $\exists b \in B,\ b \notin A$ *i.e.* $A \cap B = A \neq B$*.*

Notation 4. *I_n : $I_n \in \mathbb{R}^{n\times n}$ is the identity matrix with all diagonal coefficient equal to one and the rest set to zero. It is sometime noted $I_n = diag(1, \cdots, 1)$.*

Notation 5. *A^T : Given a matrix $A \in \mathbb{R}^{n\times m}$, $A^T \in \mathbb{R}^{m\times n}$ is the transposed matrix verifying $A^T_{ij} = A_{ji}, \forall (i,j) \in \{1, \cdots, m\} \times \{1, \cdots, n\}$.*

Notation 6. *$\mathrm{Tr}(A)$: Given a matrix $A \in \mathbb{R}^{n\times n}$, $\mathrm{Tr}(A)$ is the trace of A verifying $\mathrm{Tr}(A) = \sum_{i=1}^{n} A_{ii}$.*

Notation 7. *$GL_n(\mathbb{R})$: Set of all invertible matrix in $\mathbb{R}^{n\times n}$. Invertible means that for a matrix $A \in GL_n(\mathbb{R})$, there exist a matrix $B \in \mathbb{R}^{n\times n}$ such that $AB = BA = I_n$. Additionnaly, B is noted A^{-1} and is also in $GL_n(\mathbb{R})$.*

Notation 8. *$\mathcal{P}(\mathcal{X})$: Set of all $\mathcal{X}$-subsets. We can write : $\mathcal{P}(\mathcal{X}) = \{\mathcal{U} \,|\, \mathcal{U} \subset \mathcal{X}\}$*

Notation 9. *$||\cdot||$: denotes an usual norm for the space our element is in. As we are always in finite dimension they are all equivalent (Property 2).*

Notation 10. *$||\cdot||_2$: denotes a specific norm called euclidian norm. Computed as follows in a R^n space : $\|\cdot\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$.*

Notation 11. *$f|_K$: given a function $f : \mathcal{X} \longrightarrow \mathcal{Y}$ with $K \subset \mathcal{X}$, $f|_K : x \in K \mapsto f(x)$ is the restriction of f over the set K.*

Notation 12. *$\nabla_x f(x')$: denotes the gradient of f over the coordinate x taken on the point x'.*

Notation 13. *$\mathcal{J}_f(x)$: denotes the jacobian matrix of the function g taken on the point x.*

Notation 14. *$C^k(\mathcal{X})$: denotes the set of k-differentiable functions which are continuous on their k^{th} differential. We denote $C^0(\mathcal{X})$ the continuous function set.*

Properties :

Property:

Property 2. ***(Norm equivalence theorem).*** *In a given normed vector space over a complete field, all norms are equivalent. That is, given $\|\cdot\|_a$ and $\|\cdot\|_b$, there exist $M_a, M_b > 0$ such that*

$$M_b \|x\|_a \;\leq\; \|x\|_b \;\leq\; M_a \|x\|_a, \quad \forall x.$$

Property:

Property 3. ***($\mathcal{M}_r$ equality).*** *We define the set :*

$$\mathcal{M}_r := \{X \in \mathbb{R}^{n\times m} : \mathrm{rank}(X) \leq r\}$$

Then, $\mathcal{M}_r$ is equal to the set $\{\, X \in \mathbb{R}^{n\times m} \;:\; \exists U \in \mathbb{R}^{n\times r},\, W \in \mathbb{R}^{m\times r}$ with $X = UW^\top \,\}$.

Proof. ($\supset$) If $X = UW^\top$ then $\mathrm{rank}(X) \leq r$. Thus, we have the first inclusion.

($\subset$) Assume we have X such that $\mathrm{rank}(X) \leq r$. We can use the Thin Singular Value Decomposition (Thin SVD) in order to derive $X = U_s \Sigma_s V_s^\top$ where $U_s \in \mathbb{R}^{n\times r}$, $V_s \in \mathbb{R}^{m\times r}$ and $\Sigma_s \in \mathbb{R}^{r\times r}$. Such a Thin SVD always exist for matrix with

coefficient in $\mathbb{R}$ such that there rank is r and Σ_s is a diagona matrix with positive coefficient (Theorem 2.4.1 & Section 2.4.3 [GVL13]). This allow us to rewrite $\Sigma_s = \Sigma_s^{1/2}(\Sigma_s^{1/2})^\top$ and finally give the classical decomposition of $X = \tilde{U}_s\tilde{V}_s^T$ where $\tilde{U}_s = U_s\Sigma_s^{1/2}$ and $\tilde{V}_s = V_s\Sigma_s^{1/2}$. This ultimately prove that X is in the set $\{\, X \in \mathbb{R}^{n\times m} \;:\; \exists U \in \mathbb{R}^{n\times r},\ W \in \mathbb{R}^{m\times r} \text{ with } X = UW^\top \,\}$. □

Property:

Property 4. ***(Equivalence under compactness and continuity)***. *Let $\mathcal{J} \subset \mathbb{R}^{n\times r}$ and $\mathcal{K} \subset \mathbb{R}^{m\times r}$ be compact sets, let $\|\cdot\|$ be a norm on $\mathbb{R}^{n\times r}$, and let*

$$F : \mathcal{J}\times\mathcal{K} \to \mathbb{R}, \qquad F(U,W) := \mathcal{E}\big(A,\, UW^T\big)$$

be continuous on the compact set $\mathcal{J}\times\mathcal{K}$. For each $\lambda \geq 0$ consider the penalized problem

$$(P_\lambda) \qquad \Phi(\lambda) := \min_{(U,W)\in\mathcal{J}\times\mathcal{K}} F(U,W) + \lambda\|U\|,$$

and for each $\tau \geq 0$ consider the constrained problem

$$(Q_\tau) \qquad \Psi(\tau) := \min_{\substack{(U,W)\in\mathcal{J}\times\mathcal{K}\\ \|U\|\leq\tau}} F(U,W).$$

Then:

1. *For every $\lambda \geq 0$ and every $\tau \geq 0$ the minima in (P_λ) and (Q_τ) are attained (i.e. the minima exist).*
2. *For each $\lambda \geq 0$ let $\mathcal{M}(\lambda) \subset \mathcal{J}\times\mathcal{K}$ denote the (nonempty) set of minimizers of (P_λ) and define*

   $$\tau(\lambda) := \min\{\|U\| :\ (U,W) \in \mathcal{M}(\lambda)\}.$$

   Then the function $\lambda \mapsto \tau(\lambda)$ is well-defined and nonincreasing on $[0,\infty)$. Moreover the image

   $$\mathcal{I} := \{\, \tau(\lambda) : \lambda \geq 0 \,\}$$

   is included in a compact interval $[\tau_-, \tau_+] \subset \mathbb{R}_+$.
3. *For any $\tau \in \mathcal{I}$ there exists $\lambda \geq 0$ and a minimizer $(U^\star, W^\star) \in \mathcal{M}(\lambda)$ such that $\|U^\star\| = \tau$, and this $(U^\star, W^\star)$ is also a minimizer of the constrained problem (Q_τ). Conversely, if $(U^\star, W^\star)$ minimizes (Q_τ) and there exists $\lambda \geq 0$ for which $(U^\star, W^\star) \in \mathcal{M}(\lambda)$, then it also minimizes (P_λ).*
4. *Such results also hold when exchanging penalization (constraint) on U to penalization (constraint) on W.*

Proof. [47]

[47] Proof is a rearrangement of element from LectureNote and MITNote to our problem.

[1] For every fixed $\lambda \geq 0$, since $\mathcal{J} \times \mathcal{K}$ is compact and F is continuous, the function

$$(U, W) \mapsto F(U, W) + \lambda \|U\|$$

is continuous on a nonempty compact set. By the Weierstrass extreme value theorem [Wikipedia] there exists at least one minimizer of (P_λ). Similarly, for every $\tau \geq 0$ the set $\{(U, W) \in \mathcal{J} \times \mathcal{K} : \|U\| \leq \tau\}$ is closed and bounded (thus, compact), and F is continuous, thus (Q_τ) attains its minimum.

[2] Let us fix $\lambda \geq 0$ and let $\mathcal{M}(\lambda)$ be the nonempty set of minimizers of (P_λ). Since $\mathcal{M}(\lambda) \subset \mathcal{J} \times \mathcal{K}$ and $\mathcal{J} \times \mathcal{K}$ is compact, the set

$$\{\|U\| : \ (U, W) \in \mathcal{M}(\lambda)\}$$

is a nonempty compact subset of $\mathbb{R}_+$[48], so the minimum $\tau(\lambda)$ is properly defined.

Now, to prove monotonicity: we take $0 \leq \lambda_1 < \lambda_2$ and choose minimizers $(U_1, W_1) \in \mathcal{M}(\lambda_1)$ and $(U_2, W_2) \in \mathcal{M}(\lambda_2)$. By optimality of (U_1, W_1) for (P_{λ_1}),

$$F(U_1, W_1) + \lambda_1 \|U_1\| \leq F(U_2, W_2) + \lambda_1 \|U_2\|. \tag{1}$$

Same for (U_2, W_2) solution to (P_{λ_2}),

$$F(U_2, W_2) + \lambda_2 \|U_2\| \leq F(U_1, W_1) + \lambda_2 \|U_1\|. \tag{2}$$

Using (1) and (2) and canceling $F(U_1, W_1) + F(U_2, W_2)$, we obtain

$$\lambda_1 \|U_1\| + \lambda_2 \|U_2\| \leq \lambda_1 \|U_2\| + \lambda_2 \|U_1\|.$$

This gives us :

$$(\lambda_2 - \lambda_1)\big(\|U_2\| - \|U_1\|\big) \leq 0.$$

Since we had $\lambda_2 - \lambda_1 > 0$, we deduce $\|U_2\| \leq \|U_1\|$. Taking the minimal norms over the respective minimizer sets yields $\tau(\lambda_2) \leq \tau(\lambda_1)$. Thus $\lambda \mapsto \tau(\lambda)$ is nonincreasing.

Finally, because each $\tau(\lambda)$ lies in the compact interval (Recall that the set $\{\|U\| : (U, W) \in \mathcal{J} \times \mathcal{K}\}$ is compact)

$$[\,\min\{\|U\| : (U, W) \in \mathcal{J} \times \mathcal{K}\},\ \max\{\|U\| : (U, W) \in \mathcal{J} \times \mathcal{K}\}\,],$$

the image $\mathcal{I}$ is bounded and in $\mathbb{R}_+$, thus giving us [2].

[3] Let $\tau \in \mathcal{I}$. By definition there exists $\lambda \geq 0$ and a minimizer $(U^\star, W^\star) \in \mathcal{M}(\lambda)$ with $\|U^\star\| = \tau$. For any couple (U, W) verifying constraints of (Q_τ) (so $\|U\| \leq \tau$) we have by optimality of $(U^\star, W^\star)$

$$F(U^\star, W^\star) + \lambda \|U^\star\| \leq F(U, W) + \lambda \|U\| \leq F(U, W) + \lambda \tau,$$

[48] Because the norm is continuous.

Subtracting $\lambda\tau$ from both sides yields

$$F(U^\star, W^\star) \leq F(U, W).$$

Because this holds for every couple (U, W) verifying (Q_τ) constraint, $(U^\star, W^\star)$ is a minimum of (Q_τ). In the same way, if $(U^\star, W^\star)$ minimizes (Q_τ) and also belongs to $\mathcal{M}(\lambda)$ for some λ, then it obviously minimizes (P_λ) by definition. Thus proving [3].

[4] The proof is done over U but could completely be done with W. □

Property:

Property 5. *(Coordinate projection is 1-Lipschitz) Let $p, q \in \mathbb{N}$. We have $\mathbb{R}^{p+q}$ and $\mathbb{R}^p$ with their norms $\|\cdot\|_{\mathbb{R}^{p+q}}$ and $\|\cdot\|_{\mathbb{R}^p}$. Define the projection onto the first[a] p coordinates*

$$\pi : \mathbb{R}^{p+q} \to \mathbb{R}^p, \qquad \pi(x_1, \dots, x_p, x_{p+1}, \dots, x_{p+q}) = (x_1, \dots, x_p).$$

Then π is Lipschitz with Lipschitz constant 1, written : For all $u, v \in \mathbb{R}^{p+q}$,

$$\|\pi(u) - \pi(v)\|_{\mathbb{R}^p} \leq \|u - v\|_{\mathbb{R}^{p+q}}.$$

[a] Truthfully, we can project on any batch of coordinate. We will still have the same results of our projection being 1-Lipschitz.

Proof. Write $u = (u^{(1)}, u^{(2)})$ and $v = (v^{(1)}, v^{(2)})$ with $u^{(1)}, v^{(1)} \in \mathbb{R}^p$ (first p coordinates) and $u^{(2)}, v^{(2)} \in \mathbb{R}^q$ (last q coordinates). Then using the Euclidean norm

$$\|\pi(u) - \pi(v)\|_2^{\mathbb{R}^p} = \|u^{(1)} - v^{(1)}\|_2^{\mathbb{R}^p} = \sqrt{\sum_{i=1}^{p} (u_i - v_i)^2}.$$

On the other hand

$$\|u - v\|_2^{\mathbb{R}^{p+q}} = \sqrt{\sum_{i=1}^{p+q} (u_i - v_i)^2} = \sqrt{\sum_{i=1}^{p} (u_i - v_i)^2 + \sum_{i=p+1}^{p+q} (u_i - v_i)^2}.$$

Since the additional term is positive, we have

$$(\|u^{(1)} - v^{(1)}\|_2^{\mathbb{R}^p})^2 \leq (\|u - v\|_2^{\mathbb{R}^{p+q}})^2,$$

and taking square roots yields

$$\|\pi(u) - \pi(v)\|_2^{\mathbb{R}^p} \leq \|u - v\|_2^{\mathbb{R}^{p+q}}.$$

Thus, using Norm equivalence theorem (Property 2) we have that π is 1-Lipschitz for any norm $\|\cdot\|_{\mathbb{R}^{p+q}}$ and $\|\cdot\|_{\mathbb{R}^p}$. □

Property:

Property 6. *For $U \subset \mathbb{R}^p$ and $V \subset \mathbb{R}^q$ open set. If $h : U \longrightarrow V$ and $g : V \longrightarrow \mathbb{R}^n$ are locally lipschitz, then $g \circ h$ is also locally lipschitz over U.*

Proof. Assume we are in the setting of our property, let us note $x_0 \in U$.

We have that h is locally Lipschitz at x_0. Hence there exist $r_1 > 0$ and $L_1 \geq 0$ such that

$$\|h(y) - h(z)\| \leq L_1 \|y - z\| \qquad \text{for all } y, z \in B(x_0, r_1) \cap U. \tag{1}$$

Also, since g is locally Lipschitz at the point $h(x_0) \in V$, there exist $r_2 > 0$ and $L_2 \geq 0$ such that

$$\|g(u) - g(v)\| \leq L_2 \|u - v\| \qquad \text{for all } u, v \in B(h(x_0), r_2) \cap V. \tag{2}$$

Define

$$r := \min\left(r_1,\ \frac{r_2}{2L_1}\right).$$

(If $L_1 = 0$ then h is constant on $B(x_0, r_1)$ and any positive choice of $r \leq r_1$ will do).

Let $y, z \in B(x_0, r) \cap U$. From (1) with $z = x_0$ we get

$$\|h(y) - h(x_0)\| \leq L_1 \|y - x_0\| < L_1 r \leq L_1 \frac{r_2}{2L_1} = \frac{r_2}{2},$$

and similarly $\|h(z) - h(x_0)\| < r_2/2$. Hence both $h(y)$ and $h(z)$ lie in $B(h(x_0), r_2/2)$, and therefore in $B(h(x_0), r_2) \cap V$. By (2) we may apply the Lipschitz bound for g to the pair $u = h(y)$, $v = h(z)$:

$$\|g(h(y)) - g(h(z))\| \leq L_2 \, \|h(y) - h(z)\|.$$

Combining this with (1) (valid on $B(x_0, r)$) yields

$$\|g(h(y)) - g(h(z))\| \leq L_2 \, L_1 \, \|y - z\| \qquad \text{for all } y, z \in B(x_0, r) \cap U.$$

Thus $g \circ h$ is Lipschitz on the neighborhood $B(x_0, r) \cap U$ with Lipschitz constant $L := L_1 L_2$. Since we have this results for any $x_0 \in U$, $g \circ h$ is locally Lipschitz on U. □

Property:

Property 7. *Let $U \subset \mathbb{R}^n$ open set.*

1. *If $f, g : U \to \mathbb{R}^p$ are locally Lipschitz, then $f + g$ and fg are locally Lipschitz on U.*
2. *If $f_1, \ldots, f_k : U \to \mathbb{R}^p$ are locally Lipschitz and $a_1, \ldots, a_k \in \mathbb{R}$, then the linear combination $\sum_{i=1}^k a_i f_i$ is locally Lipschitz.*
3. *If $f : U \to \mathbb{R}^p$ and $g : U \to \mathbb{R}^q$ are locally Lipschitz, then the concatenation map $c : U \to \mathbb{R}^{p+q}$, $c(x) := (f(x), g(x))$ is locally Lipschitz on U.*

Proof. Let $x_0 \in U$. Since f and g are locally Lipschitz at x_0, there exists $r > 0$ and constants $L_f, L_g \geq 0$ such that for all $y, z \in B(x_0, r) \cap U$,

$$\|f(y) - f(z)\| \leq L_f \|y - z\|, \qquad \|g(y) - g(z)\| \leq L_g \|y - z\|. \tag{3}$$

Sum - For $y, z \in B(x_0, r) \cap U$,

$$\|(f+g)(y) - (f+g)(z)\| \leq \|f(y) - f(z)\| + \|g(y) - g(z)\| \leq (L_f + L_g)\|y - z\|.$$

Thus $f + g$ is Lipschitz on $B(x_0, r) \cap U$ with constant $L_f + L_g$.

Product - First we need boundedness on the ball : From (3) with $z = x_0$ we get for every $y \in B(x_0, r) \cap U$,

$$\|f(y)\| \leq \|f(x_0)\| + L_f\|y - x_0\| \leq \|f(x_0)\| + L_f r,$$

and similarly $\|g(y)\| \leq \|g(x_0)\| + L_g r$. Hence the restrictions of f and g to $B(x_0, r) \cap U$ are bounded; set

$$M_f := \sup_{x \in B(x_0,r)\cap U} \|f(x)\| \leq |f(x_0)| + L_f r, \qquad M_g := \sup_{x \in B(x_0,r)\cap U} \|g(x)\| \leq |g(x_0)| + L_g r.$$

For $y, z \in B(x_0, r) \cap U$ use the identity

$$f(y)g(y) - f(z)g(z) = f(y)\big(g(y) - g(z)\big) + g(z)\big(f(y) - f(z)\big),$$

and estimate

$$\|f(y)g(y) - f(z)g(z)\| \leq M_f \, \|g(y) - g(z)\| + M_g \, \|f(y) - f(z)\| \leq \big(M_f L_g + M_g L_f\big)\|y - z\|.$$

Thus, fg is Lipschitz on $B(x_0, r) \cap U$ with constant $M_f L_g + M_g L_f$.

Linear combinations and finite sums/products - This derives from the two precedent results. Indeed, constant function are locally lipschitz, thus linear combinations of locally lipschitz function are products and sum of locally lipschitz function, which means we can apply our precedent results by recurrence.

Concatenation - We first show this results using euclidean norm : For $y, z \in B(x_0, r) \cap U$ we have

$$c(y) - c(z) = \big(f(y) - f(z),\, g(y) - g(z)\big) \in \mathbb{R}^{p+q},$$

and thus by the Euclidean norm on $\mathbb{R}^{p+q}$,

$$\|c(y) - c(z)\|_2^{\mathbb{R}^{p+q}} = \sqrt{(\|f(y) - f(z)\|_2^{\mathbb{R}^p})^2 + (\|g(y) - g(z)\|_2^{\mathbb{R}^q})^2}.$$

Using (3) we obtain

$$\|c(y) - c(z)\|_2^{\mathbb{R}^{p+q}} \leq \sqrt{(L_f\|y - z\|_2^{\mathbb{R}^n})^2 + (L_g\|y - z\|_2^{\mathbb{R}^n})^2} = \sqrt{L_f^2 + L_g^2}\, \|y - z\|_2^{\mathbb{R}^n}.$$

Hence c is Lipschitz on the neighbourhood $B(x_0, r) \cap U$ with Lipschitz constant $L := \sqrt{L_f^2 + L_g^2}$.

Then, using Property 2 (norm equivalence) : For any norm $\|\cdot\|$ on $\mathbb{R}^{p+q}$ we have that there exist $M > 0$ such that $\|c(y) - c(z)\| \leq M \, \|c(y) - c(z)\|_2^{\mathbb{R}^{p+q}} \leq M \sqrt{L_f^2 + L_g^2}\, \|y - z\|_2^{\mathbb{R}^n}$. Which give us our expected results. □

Property:

Property 8. *Given σ, an activation function that is locally lipschitz. Given $x \in \mathcal{X} \subset \mathbb{R}^n$ (with $\mathcal{X}$ bounded). Then a transformer (See Def 5) using σ is also locally lipschitz over its parameters.*

Proof. First, note that any C^1 function is locally lipschitz (See [MathStack]). Then we have that the $softmax$ function is C^1 (In [Medium], the jacobian of $softmax$ is given, and it is continuous). Furthermore, linear application are parameter-wise locally lipschitz (and even globally). One critical problem would be the first layer, linear embedding of x. Let us prove that it is also locally lipschitz :

We know that x is bounded. Hence, using triangular inequality, Cauchy-Schwarz and Property 5 on the first embedding layer $(W_1, b_1) \mapsto W_1 \cdot x + b_1$ we get (with euclidean norm) :

$$\begin{aligned}
\|W_1 x + b_1 - (W_1' x + b_1')\|_2 &= \|(W_1 - W_1')x + (b_1 - b_1')\|_2 \\
&\leq \|(W_1 - W_1')x\|_2 + \|b_1 - b_1'\|_2 \\
&\leq \|W_1 - W_1'\|_2 \, \|x\|_2 + \|b_1 - b_1'\|_2 \\
&\leq (1 + \|x\|_2) \, \|(W_1, b_1) - (W_1', b_1')\|_2.
\end{aligned}$$

Finally, using Property 7, we get that a transformer is locally lipschitz over its own parameter. □

Property:

Property 9. *Let $U \subset \mathbb{R}^n$ open set. If $f : U \longrightarrow \mathbb{R}^p$ is locally lipschitz and there exist $K \subset U$ a compact[a] set, then $f|_K$ is lipschitz.*

[a] We are in finite dimension, thus a compact set is equivalent to a closed and bounded set.

Proof. See [MathStack]. The proof relies on using Borel-Heine characterization of compactness. It allows to obtain a finite collection of ϵ-ray ball that completely cover our compact. Hence, as we have a locally lipschitz function for fixed $\epsilon > 0$, we can deduce that the function is globally lipschitz over the compact with global lipschitz constant being the max of the local lipschitz constant. □

8.4 Annex B: Metric mathematical definition

8.4.1 Annex B.1 : Norm Based Metric

Definition:

Definition 40. ***Frobenius Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m \times r}$, define the Frobenius distance metric $m : \mathbb{R}^{m \times r} \times \mathbb{R}^{m \times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \|W - W'\|_F = \mathrm{Tr}((W - W')(W - W')^\top) = \sqrt{\sum_{i,j} (W_{ij} - W'_{ij})^2}$$

Definition:

Definition 41. ***Spectral Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$, define the spectral distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \|W - W'\|_2 = \sigma_{\max}(W - W')$$

where $\sigma_{\max}$ denotes the largest singular value.

Definition:

Definition 42. ***Nuclear Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$, define the nuclear distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \|W - W'\|_* = \sum_i \sigma_i(W - W')$$

where σ_i are the singular values of $(W - W')$.

Definition:

Definition 43. ***Entry-wise p-Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$ and $p \geq 1$, define the entry-wise[a] p-distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \left(\sum_{i,j} |W_{ij} - W'_{ij}|^p \right)^{1/p}$$

[a] The term entry-wise here highligh the opposition with the Schatten p-distance, an other generalization of L_p norm to matrix operator.

Definition:

Definition 44. ***Procrustes Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$, define the Procrustes distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \min_{Q:Q^TQ=I} \|W - W'Q\|_F$$

where the minimum is taken over all orthogonal matrices Q.

Definition:

Definition 45. ***Schatten p-Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$ and $p \geq 1$, define the Schatten p-distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \left(\sum_i \sigma_i(W - W')^p \right)^{1/p}$$

where σ_i are the singular values of $(W - W')$.

Definition:

Definition 46. ***Matrix Logarithm Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$ and regularization $\epsilon > 0$, define the matrix logarithm distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \| \log(G_W) - \log(G_{W'}) \|_F$$

where $G_W = WW^T + \epsilon I$.

Definition:

Definition 47. ***Grassmannian Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$, define the Grassmannian distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \| \sin(\Theta) \|_F$$

where Θ contains the principal angles between $col(W)$ and $col(W')$.

Definition:

Definition 48. ***Bures Distance :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$ and regularization $\epsilon > 0$, define the Bures distance metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to \mathbb{R}_+$ as:*

$$m(W, W') = \sqrt{tr(G_W + G_{W'} - 2(G_W^{1/2} G_{W'} G_W^{1/2})^{1/2})}$$

where $G_W = WW^T + \epsilon I$.

8.4.2 Annex B.2 : Correlation metric

Definition:

Definition 49. ***Concept Columnwise correlation :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$, define the columnwise correlation metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to [0, 1]$ as:*

$$m(W, W') = \frac{1}{r} \sum_{j=1}^{r} \left| \rho(W_{\cdot j}, W'_{\cdot j}) \right|$$

where $\rho(W_{\cdot j}, W'_{\cdot j}) = \frac{W_{\cdot j} \cdot W'_{\cdot j}}{\|W_{\cdot j}\|_2 \, \|W'_{\cdot j}\|_2}$ denotes the vector correlation between.

Definition:

Definition 50. ***Concept Optimal correlation :*** *Given two matrices $W, W' \in \mathbb{R}^{m\times r}$, define the optimal correlation metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \to [0,1]$ as:*

$$m(W, W') = \frac{1}{r}\sum_{i=1}^{r} \left|\rho(W_{\cdot i}, W'_{\cdot\sigma(i)})\right|$$

where ρ is the vector correlation and σ is the optimal column assignment (permutation) minimizing the cost matrix $C_{ij} = -|\rho(W_{\cdot i}, W'_{\cdot j})|$.

Definition:

Definition 51. ***Concept Internal optimal correlation :*** *Given a matrix $W \in \mathbb{R}^{m\times r}$ with $r \geq 2$, define the internal optimal correlation metric $m : \mathbb{R}^{m\times r} \to [0,1]$ as:*

$$m(W) = \frac{1}{r}\sum_{i=1}^{r} \left|\rho(W_{\cdot i}, W_{\cdot\sigma(i)})\right|$$

where ρ is the vector correlation, and σ is the optimal permutation over distinct columns $i \neq j$ minimizing $C_{ij} = -|\rho(W_{\cdot i}, W_{\cdot j})|$.

Definition:

Definition 52. ***Concept Weighted optimal correlation :*** *Given matrices $W, W' \in \mathbb{R}^{m\times r}$ and a weight matrix $U \in \mathbb{R}^{n\times r}$, define the weighted optimal correlation metric $m : \mathbb{R}^{m\times r} \times \mathbb{R}^{m\times r} \times \mathbb{R}^{n\times r} \to [0,1]$ as:*

$$m(W, W', U) = \left(\sum_{j=1}^{n} u_j \left|\rho(W_{\cdot j}, W'_{\cdot\sigma(j)})\right|\right)$$

where $u_j = \frac{\sum_{i=1}^{k} |U_{ij}|}{\sum_{j=1}^{n}\sum_{i=1}^{k} |U_{ij}|}$ is the normalized column-wise weight and σ is the optimal column permutation from the optimal correlation metric.

8.5 Annex C: Vapnik Generalization Bound

Definition:

Definition 53. ***Vapnik-Chervonenkis Dimension (VC-dim)*** *- Let $\mathcal{H}$ be a hypothesis class of binary functions $h : \mathcal{X} \to \{0,1\}$. A set $C = \{x_1, \ldots, x_n\} \subset \mathcal{X}$ is said to be* shattered *by $\mathcal{H}$ if for every labeling $y \in \{0,1\}^n$, there exists $h \in \mathcal{H}$ such that $h(x_i) = y_i$ for all $i = 1, \ldots, n$.*

$$\mathrm{VC}(\mathcal{H}) = \max\left\{n \in \mathbb{N} \ : \ \max_{x_1,\ldots,x_n\in\mathcal{X}} |\{(h(x_1), \ldots, h(x_n)) \ : \ h \in \mathcal{H}\}| = 2^n\right\}$$

The VC dimension *of $\mathcal{H}$, denoted $\mathrm{VC}(\mathcal{H})$, is the cardinality of the largest set $C \subset \mathcal{X}$ that can be shattered by $\mathcal{H}$. If arbitrarily large sets can be shattered, then $\mathrm{VC}(\mathcal{H}) = \infty$.*

Remark:

Remark 9 (Intuition on VC Dimension)**.** *If a model class has VC–dimension d, it means:*

- ***Flexibility:*** *For* any *choice of d points in the input space, no matter how you label ("color") them red or blue, there is always some models in the class that perfectly separates the reds from the blues.*
- ***Capacity measure:*** *d is the* maximum *number of points you can* shatter*-i.e. label arbitrarily and still fit exactly. If you try $d+1$ points, there will be at least one labeling you cannot realize.*
- ***Complexity trade-off:*** *A larger VC–dimension means greater expressive power (you can fit more intricate patterns), but also a higher risk of overfitting random fluctuations in the data.*

Conclusion: *The VC–dimension d serves as a precise "flexibility score" for a model class-quantifying how complex a set of functions you can learn before you inevitably lose the ability to generalize.*

Theorem:

Theorem 1. ***(Vapnik Generalization Bound)*** *- Let $\mathcal{H}$ be a hypothesis class of binary classifiers with VC-dimension d. Let $S = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a sample of size n drawn i.i.d. from an unknown distribution $\mathcal{D}$ over $\mathcal{X} \times \{0, 1\}$. Then, there exist $C \in \mathbb{R}^+$ such that for any $\delta \in (0, 1)$, with probability at least $1 - \delta$, the following holds for all $h \in \mathcal{H}$:*

- *If $d < 2n$:*

$$|R(h) - \hat{R}_S(h)| \leq C\sqrt{\frac{d\,log\left(\frac{2en}{d}\right) + log\left(\frac{1}{\delta}\right)}{n}},$$

- *If $d > 2n$:*

$$|R(h) - \hat{R}_S(h)| \leq C\sqrt{1 + \frac{log\left(\frac{1}{\delta}\right)}{n}},$$

where $R(h)$ is the true risk or error, and $\hat{R}_S(h)$ is the empirical risk on the sample S.

Proof. Refer to Vapnik's "Statistical Learning Theory" book if you are interested [Vap98]. □

Remark:

Remark 10. ***What does Vapnik bound tell us ?*** *Vapnik bounded the difference between empirical error and objective error. This bound had huge impact on how learning theory was first constructed. Indeed, the bound implies that if $d = o(n)$ (If the complexity of the class divided by the dataset cardinal tend to zero) then for properly chosen $\delta(n) = \frac{1}{n}$ we have convergence of the empirical risk to the objective one when data size n increases. Unfortunately, this bound does not tell us anything on the case where the complexity of the class explode (The case of d increasing quicker or similarly to $2n$). Traditionally in statistical learning, thanks to Vapnik bound, it was believed that a model class with high complexity would overfit and under perform compared to one with more proper complexity. The new era of large model proved this intuition wrong.*

Theorem:

Theorem 2. *(Anthony & Bartlett, 1999 [AB99]) Let $\mathcal{H}$ be the class of functions computed by a feed-forward neural network with*

- *W real-valued weights (including biases),*
- *L layers,*
- *and binary (threshold) activation at each hidden unit.*

Then there exist constants $c_1, c_2 > 0$ such that

$$c_1 W^2 \;\leq\; \mathrm{VC}(\mathcal{H}) \;\leq\; c_2 W^2 .$$

In particular, $\mathrm{VC}(\mathcal{H}) = \Theta(W^2)$.

Let us use Theorem 1 & 2 to make an idea of how Vapnik bound is not the correct way to go for modern deep learning theory[49]. According to reporters [Blog], GPT-4, OpenAI most performant model is made of 2 trillion parameters and was trained on 13 trillion tokens (even if the number are not exactly correct, we can still assume similar scale). Using Theorem 2, we have a VC-dim $d = \Theta((10^{14})^2) = \Theta(10^{28})$ while $n = \Theta(10^{15})$. Then with Theorem 1, we see that we are clearly in the case of $d > 2n$ and Vapnik bound is irrelevant. However, GPT-4 did converge toward proper performant model.

8.6 Annex D: Proving Locally Lipschitz Continuity

8.6.1 Annex D.1 : Convex NMF Algorithm is Locally Lipschitz

We consider the Convex NMF algorithm that, given an input matrix A, produces a decomposition $(U^*(A), W^*(A))$, as described in Algorithm 5.

In the classical implementation, the algorithm also includes a tolerance-based stopping criterion. Here, we assume instead that the algorithm always runs for a fixed number of iterations $T_{\max}$, i.e., the maximum iteration count is reached before any tolerance-based stopping. Under this assumption and with a fixed initialization seed, the algorithm defines a deterministic map

$$A \mapsto (U^*(A), W^*(A))$$

which is locally Lipschitz.

Indeed, each iteration of the multiplicative-update scheme consists of a finite composition of smooth (C^1) maps with respect to A. This holds in particular because of the stabilizer $\varepsilon > 0$, which ensures that the division operation $x \mapsto \frac{1}{x+\varepsilon}$ remains $C^1(\mathbb{R}_+)$ and prevents singularities. The elementwise product and square root operations are also C^1 on $\mathbb{R}_+$. Hence, each update step is locally Lipschitz with respect to its inputs.

By Property 6, the composition of locally Lipschitz mappings is locally Lipschitz. Consequently, after a finite number of iterations, the overall map $A \mapsto (U^*(A), W^*(A))$ remains locally Lipschitz.

Conversely, if the maximum number of iterations were not fixed (e.g., depending on a tolerance criterion), the number of

[49] Obviously, both theorem are valid only for binary classification. However, I will use the example of modern LLMs to show the absurdity of using such a results for modern deep learning. Do keep in mind that this is not directly applicable because of LLMs non-binary tasks.

Algorithm 5: Convex Nonnegative Matrix Factorization (multiplicative-update solver)

Input : Data matrix $A \in \mathbb{R}^{n \times m}$ (rows = samples);
Number of concepts k;
Maximum iterations T;
Boolean `strict_convex`;
Numerical stabilizer $\varepsilon > 0$.

Output: Codes $U \in \mathbb{R}_+^{n \times k}$ and coefficients $H \in \mathbb{R}_+^{k \times n}$;
Dictionary $W = HA \in \mathbb{R}^{k \times m}$.

Initialize $U^{(0)} \geq 0 \in \mathbb{R}^{n \times k}$ and $H^{(0)} \geq 0 \in \mathbb{R}^{k \times n}$

if *strict_convex* **then**

$H_{i:}^{(0)} \leftarrow H_{i:}^{(0)} / \left(\sum_j H_{ij}^{(0)} + \varepsilon \right)$

for $t \leftarrow 0$ **to** $T - 1$ **do**

$M \leftarrow AA^\top$

$M^+ \leftarrow \frac{|M| + M}{2}$

$M^- \leftarrow \frac{|M| - M}{2}$

$UH \leftarrow U^{(t)} H^{(t)}$

$\text{num}_U \leftarrow M^+ H^{(t)\top} \; + \; (UH)\, M^- H^{(t)\top}$

$\text{den}_U \leftarrow M^- H^{(t)\top} \; + \; (UH)\, M^+ H^{(t)\top}$

$U^{(t+1)} \leftarrow U^{(t)} \odot \sqrt{\frac{\text{num}_U}{\text{den}_U + \varepsilon} + \varepsilon}$

$HUU^\top \leftarrow H^{(t)}\, U^{(t+1)}\, U^{(t+1)\top}$

$\text{num}_H \leftarrow U^{(t+1)\top} M^+ \; + \; (HUU^\top)\, M^-$

$\text{den}_H \leftarrow U^{(t+1)\top} M^- \; + \; (HUU^\top)\, M^+$

$H^{(t+1)} \leftarrow H^{(t)} \odot \sqrt{\frac{\text{num}_H}{\text{den}_H + \varepsilon} + \varepsilon}$

if *strict_convex* **then**

$H_{i:}^{(t+1)} \leftarrow H_{i:}^{(t+1)} / \left(\sum_j H_{ij}^{(t+1)} + \varepsilon \right)$

return $U \leftarrow U^{(T)}, \quad H \leftarrow H^{(T)}, \quad W \leftarrow HA$

performed iterations could vary with A, effectively yielding different algorithmic mappings. In such a case, local Lipschitz continuity would no longer be guaranteed.

8.6.2 Annex D.2 : EuroBERT is locally lipschitz

Let θ denote the finite parameter vector collecting all weights. We study the deterministic forward map $f(\theta)$ for a fixed input X (evaluation mode, no dropout).

What EuroBERT changes ? Compared to the original Vaswani et al. encoder layer, EuroBERT (implementation considered) replaces or adopts the following engineering choices:

1. **Grouped-Query Attention (GQA)**: fewer K, V heads than Q heads (memory saving).
2. **Rotary Positional Embedding (RoPE)**: apply per-position pairwise rotations to Q, K instead of additive sinusoidal encodings.
3. **RMSNorm & pre-norm ordering**: use RMSNorm before sublayers (with $\varepsilon > 0$).
4. **Gated FFN (SwiGLU)**: use a gated activation (SiLU) inside the MLP.

Local Lipschitzness (short proofs) We show each layer is locally Lipschitz in parameters (for fixed input length and fixed input token ids).

- **Linear maps (embeddings, projections, output matrices).** These are affine/linear in their weight matrices; linear maps are globally Lipschitz in parameters (operator norm bound).
- **RoPE.** For fixed position p and fixed frequency schedule, RoPE_p is linear: $\mathrm{RoPE}_p(v) = R(p)v$ with orthogonal $R(p)$. Hence it is C^∞ and globally Lipschitz in v with constant 1. When $v = v(\theta)$ depends on parameters and $v(\theta)$ is locally Lipschitz, composition preserves local Lipschitzness:

$$\|\,\mathrm{RoPE}(v(\theta_1)) - \mathrm{RoPE}(v(\theta_2))\| \le \|v(\theta_1) - v(\theta_2)\|.$$

  If the frequency parameters are *learned* smoothly, RoPE is jointly C^∞ in (v, freq) and thus locally Lipschitz.
- **repeat_kv.** This is a linear replication operator $\mathcal{R}$ with $\|\mathcal{R}\|_{\mathrm{op}} = \sqrt{r}$ (Frobenius norm). Linear $\Rightarrow$ globally Lipschitz with constant $\sqrt{r}$.
- **Softmax and attention.** The pre-softmax scores are finite polynomials in parameters (bilinear combinations of projections), hence C^∞. Softmax is C^∞ on $\mathbb{R}^n$ (denominator > 0). Composition is $C^\infty \Rightarrow$ locally Lipschitz.
- **RMSNorm.** With $\varepsilon > 0$, the denominator $\sqrt{\frac{1}{d}\sum x_i^2 + \varepsilon}$ is strictly positive and RMSNorm is smooth in (x, γ). In particular RMSNorm is C^∞ and thus locally Lipschitz in parameters (scale γ) and in any smoothly varying $x(\theta)$.
- **SwiGLU (SiLU gating).** SiLU is smooth (C^∞); linear projections and pointwise multiplication preserve smoothness. Hence SwiGLU is C^∞ and locally Lipschitz.

Composition and conclusion Each layer is a finite composition and sum of locally Lipschitz (indeed at least C^1) maps, so each layer is locally Lipschitz in θ. A finite stack of such layers (the full EuroBERT forward $f(\theta)$ for fixed finite input) is locally Lipschitz (Property 6).

8.7 Annex E: Not So Short Look At Concept

8.7.1 Annex E.1 : Concept PCA visual

Before being a mechanistic interpretability extractor, Principal Component Analysis is a method to produce visual. It makes the assumptions that we can represent the data as a linear manifold of r-dimension (in our case, we make 2D plots). Hence, we will use PCA visualization to observe concept repartition and evolution across time (*ATTENTION*: We are now fixing the layers and we use colored-gradient upon the time step). See Figure 11.

What does PCA tell us about the concept manifold? Apart from the very interesting figure that PCA shows us, we see that the learning process goes along a nonlinear manifold. This implies that a more local study is necessary in order to assess proper observation on our concept manifold.

8.7.2 Annex E.2 : Concept Equivalence

Not only can we compare concept retrieved but we can also compare the space representation itself (their matrix form at least). To study the space evolution of basis change, we will define an equivalence relationship :

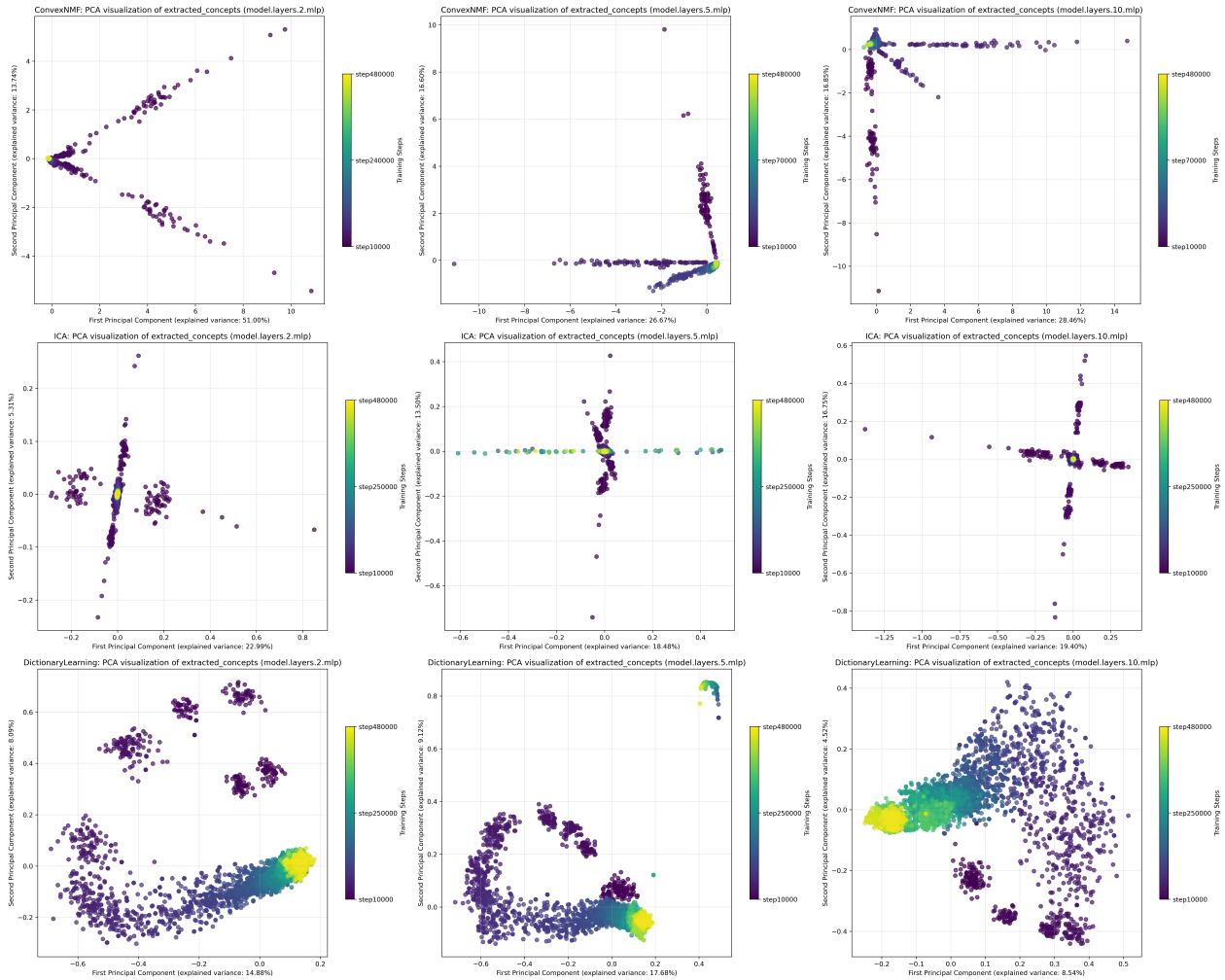

Figure 11: **Concepts PCA visual** of EuroBERT-210m over 480k steps with gradient-colored time step curves. Layers (left→right): 2^{nd} MLP, 5^{th} MLP and 10^{th} MLP of transformer block. Concepts (up→down): Convex NMF (See Def 19), ICA (See Def 27) and Dictionary Learning (See Def 29).

Definition:

Definition 54. ***Invertible Transform Equivalence*** - *Let $U \in \mathbb{R}^{n\times r}$ and $W \in \mathbb{R}^{m\times r}$ be factor matrices, and let $X = UW^\top \in \mathbb{R}^{n\times m}$ be their product. We define the* invertible equivalence relation $\sim_{\text{Inv}}$ *on pairs (U, W) by*

$$(U, W) \sim_{\text{Inv}} (U', W') \iff \exists R \in GL_r(\mathbb{R}) \text{ such that } U' = UR \quad \text{and} \quad W' = WR^{-T}.$$

The equivalence class of (U, W) under $\sim_{\text{Inv}}$ is denoted

$$[(U, W)]_{\text{Inv}} = \{(UR, WR^{-T}) : R \in GL_r(\mathbb{R})\}.$$

This is indeed an equivalence relationship:

- Obviously, it is **reflexive** using identity matrix.
- It is **symmetric** thanks to the matrix being invertible (We take for the other couple the inverse of the equivalence matrix).

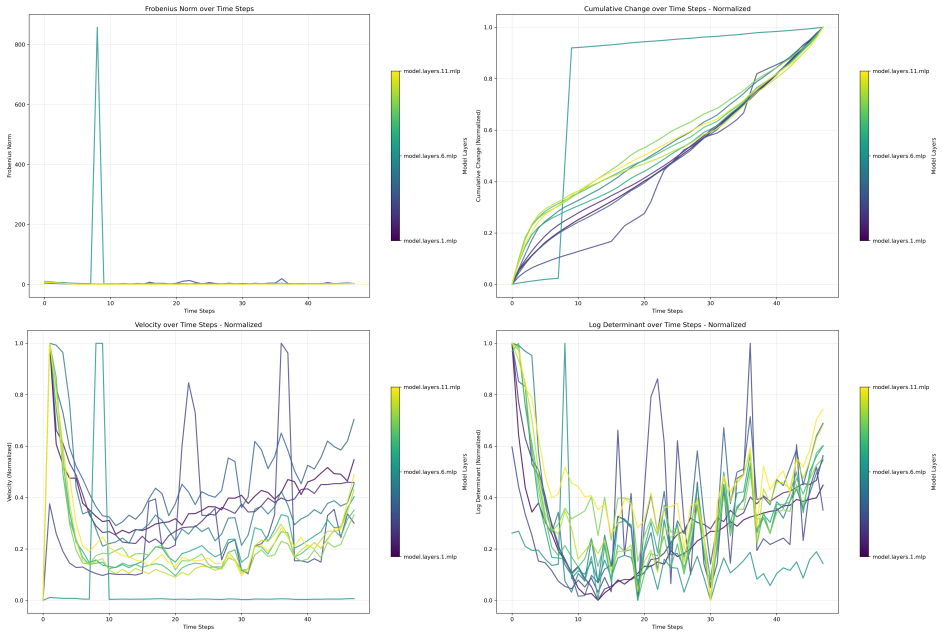

Figure 12: **Transform Matrix Evolution** over 480k step quantified with (left→right) Frobenius reconstruction error, cumulative change, velocity and log-determinant.

- It is also **transitive** because the product of invertible matrix is also invertible.

Because our methods presented in Section 4 have non-unique solution but still structured one (for most). We want to study the transformation of the space itself :

Given that we have the extracted matrix (U_T, W_T) at the final time step T and an intermediary step t extracted matrix (U_t, W_t), we compute the transform matrix $\mathcal{T}_t$:

$$\mathcal{T}_t = \arg\min_{\mathcal{T} \in GL_r(\mathbb{R})} \|W_t - W_T\mathcal{T}\|_F, \text{ subjected to } \|U_t - U_T\mathcal{T}^{-1}\|_F$$

In practice, we only search for the matrix with $r \times r$ dimension minimizing the Frobenius norm for the concept matrix W. Then, we compute the reconstruction error $Err : \mathcal{T} \mapsto \|U_t - U_T\mathcal{T}^{-1}\|_F$. It is in fact almost[50] impossible to found a non-invertible matrix for $\mathcal{T}$, in case the matrix is computationally non-invertible, one can use the Moore-Penrose pseudo-inverse. Nonetheless, in our case we retrieve such a transform $(\mathcal{T}_t)$ for each step and layer. Then, we plot the following metric :

- **Frobenius Error:** $Err_t = \|U_t - U_T\mathcal{T}_t^{-1}\|_F$
- **Cumulative Change:** $C_t = \sum_{k=1}^{t} \|\mathcal{T}_k - \mathcal{T}_{k-1}\|_F$
- **Velocity:** $\mathcal{V}_t = \|\mathcal{T}_t - \mathcal{T}_{t-1}\|_F$
- **Log-Determinant:** $\ell_t^{det} = Log|det(\mathcal{T}_t)|$

This gives us Figure 12 : The spike we observe in the reconstruction error is in the 6^{th} layer, an irregular one. It is mostly due to the fact that the computed transform is near non-invertible (exploding its inverse matrix).

For further visualization on the concept space evolution, you can find some animation video we have done here : [YouTube]

[50] The term almost here holds a mathematical meaning : A statement is said to hold almost everywhere if the set where it does not hold is of Lebesgue measure zero.

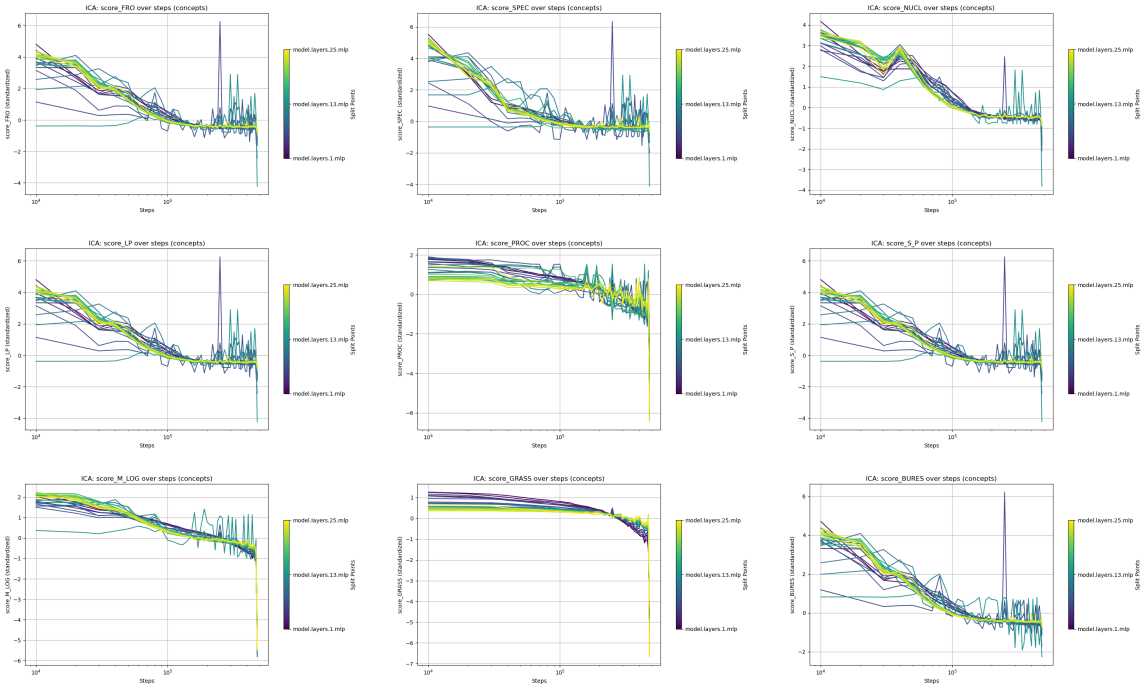

Figure 13: **Empirical Learning Dynamics** of EuroBERT-610m over 480k steps with gradient-colored layer log-curves (25 transformer layers). Let ϕ be the **ICA** extractor (Def. 27). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

Interpretation: For most layers, we once again observe this phenomenon of double regime. However, when smoothed (Cumulative Change) the second regime instability disappears. This implies that we have a stochastic behavior that has much lower impact on the concept space transformation[51]. This would go in the same direction has what we have observed with the encoding method.

8.8 Annex F: Other Plot

8.8.1 Annex F.1 : Learning Dynamic for EuroBERT-610m

See Figure 13 : EuroBERT-610m is a much larger model than the 210m. However, its learning dynamic behave similarly with the 210m. We did not study more on the difference between both since we have not found notable difference in their behavior.

One point that we can make is that EuroBERT-610m also have irregular layers which have unstable & non-convergent behavior. Once again, those layers are near middle one.

8.8.2 Annex F.2 : Correlation Plot

See Figure 14 & 15 : We plot correlation according to 8.4.2. Correlation is not a norm in itself, but it is a locally lipschitz function under assumption that we do not have zero valued concepts vectors. The function $r \mapsto 1 - r$ applied to our different correlation values behave similarly than other metric we have plotted before. In general, changing the metric

[51] Recall cumulative change definition and not that here we have a normalized version, a stable increase means that the stochastic regime does not play such a huge part in the cumulative change.

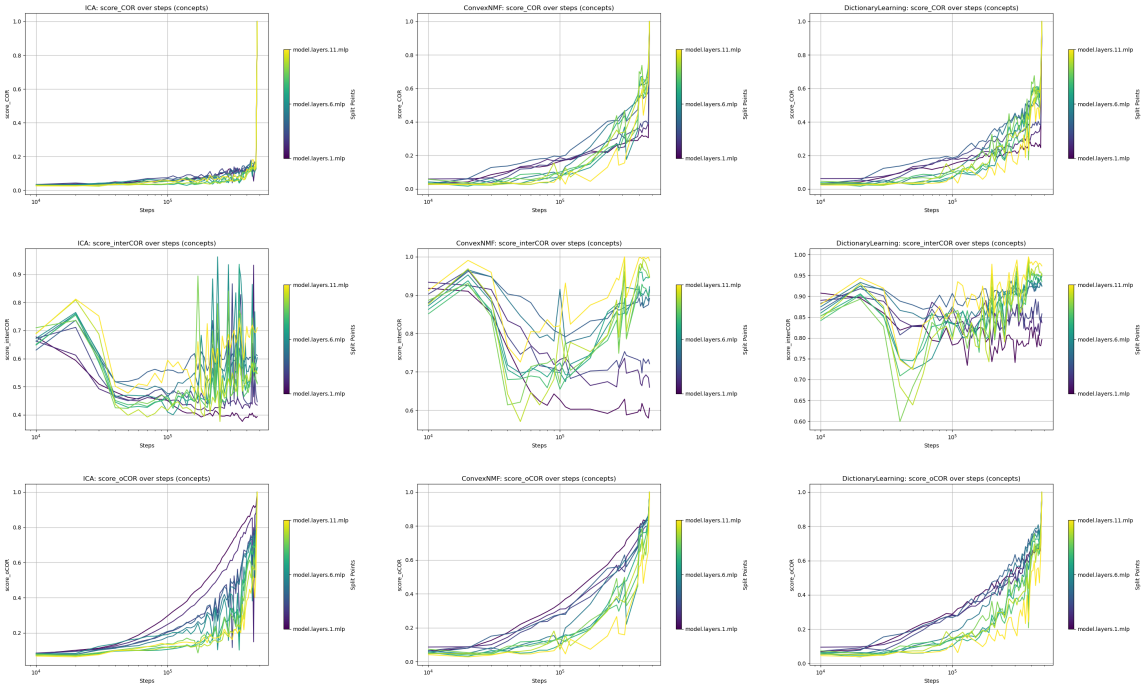

Figure 14: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be (left→right) the **ICA, ConvexNMF** and **Dictionary Learning** extractor. Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (up→down): Correlation (COR), Internal Correlation (InterCOR) and Optimal Correlation (oCOR)[Sec. 8.4.2].

for some other continuous one will not improve our observation. That is why we changed the usual mechanistic method itself. Nonetheless, ICA weighted correlation give a clearer distinction between the two regimes we observed.

8.8.3 Annex F.3 : Exponential Fitting

As it appears that learning dynamics converge with linear convergence during their first regime, we try to fit an exponential curves onto these curves using simple MLP (searching parameter (A, c) such that $t \mapsto Ae^{ct}$ fit the curves). See Figure 16.

8.8.4 Annex F.4 : SAE Learning Dynamic

As mentionned before, SAE is a highly sparse methods. Which make it less coherent for our study. Indeed, even if it successfully extract features, there is little chance that most non-zero valued features will be the same. This phenomenon has been observed for vision task too [PSF+25]. Here, we provide the SAE learning dynamics to motivate our decision : See Figure 17.

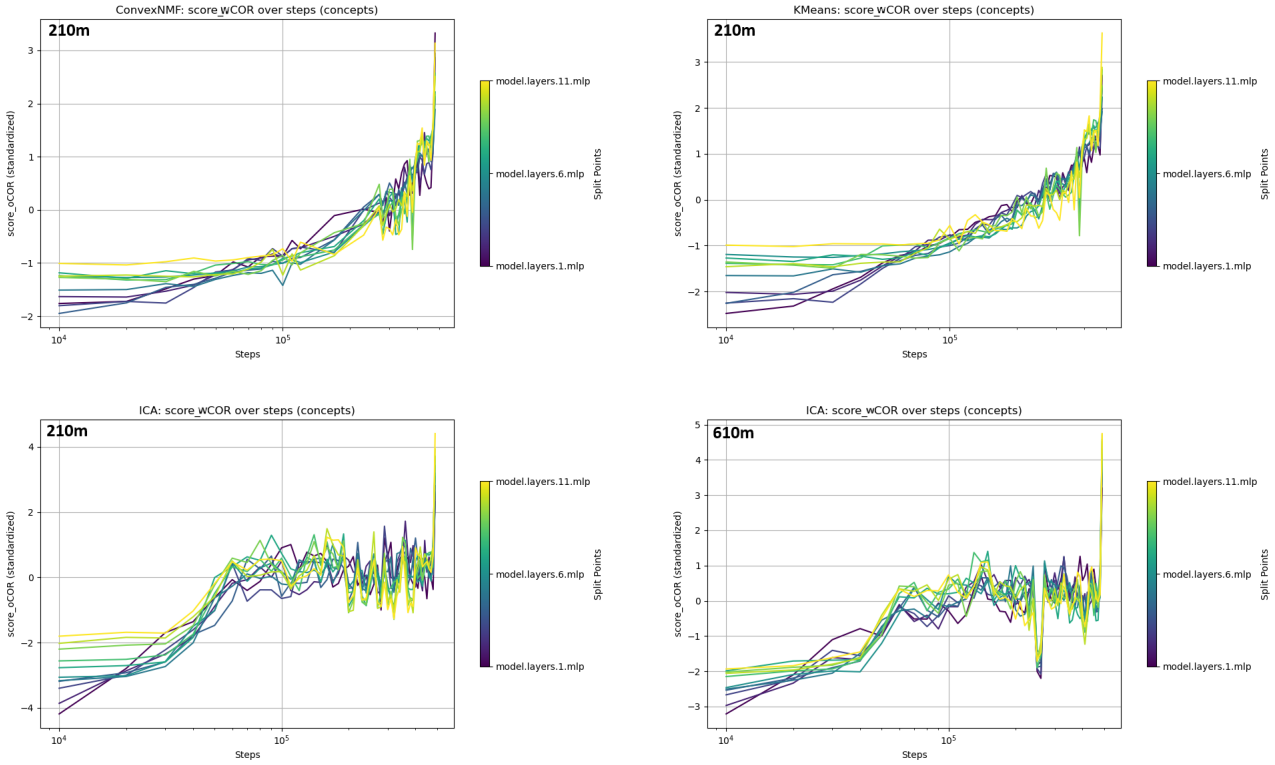

Figure 15: **Empirical Learning Dynamics** of EuroBERT-210m/610m over 480k steps with gradient-colored layer log-curves (11 transformer layers). For **ConvexNMF, KMeans** and **ICA** extractor. Each dynamic uses the **Weighted Correlation** [Sec. 8.4.2].

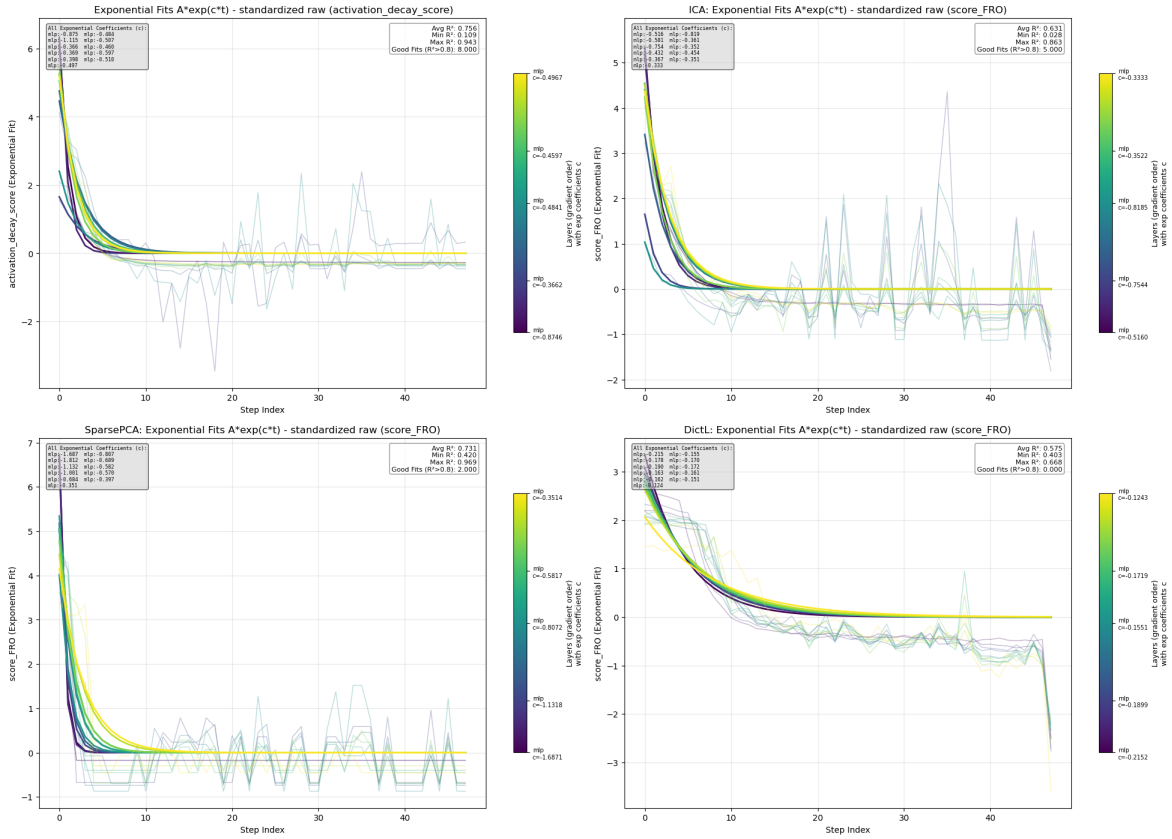

Figure 16: **Estimated Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer curves (11 transformer layers). For **Activation decay, ICA, Sparse PCA** and **Dictionary Learning**. Each dynamic is fitted to an exponential curves (Coefficient is provided from up to down, left to right). R^2 is the coefficient of determination : Nearer it is from 1, the better is the fitted model.

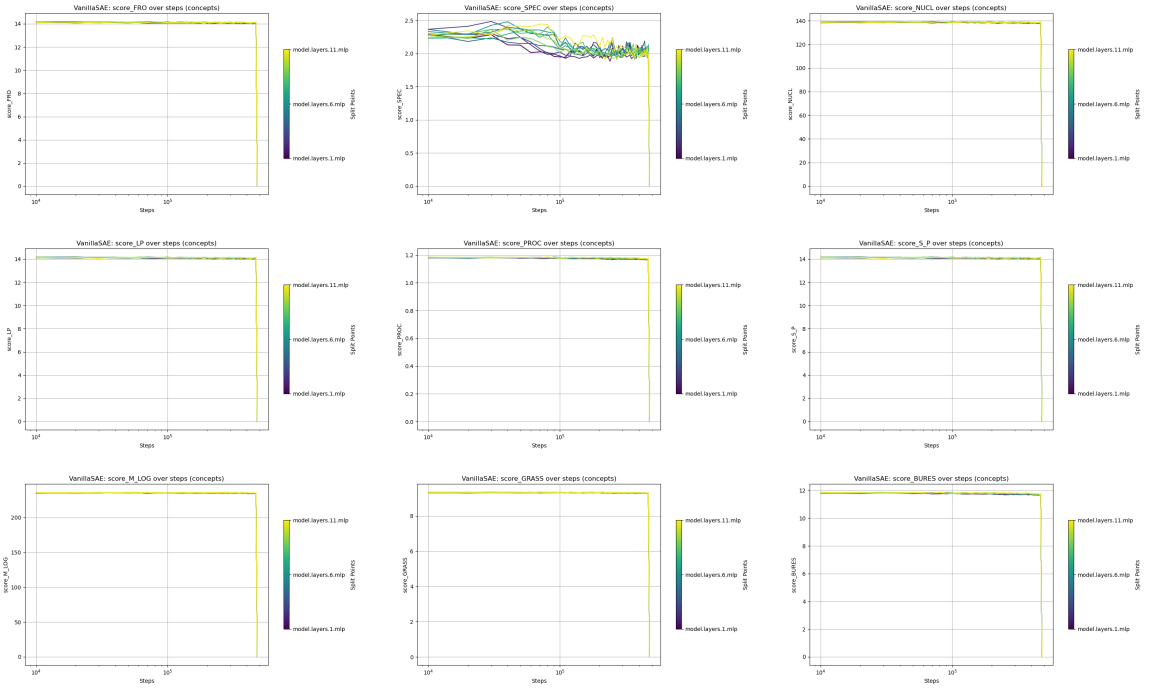

Figure 17: **Empirical Learning Dynamics** of EuroBERT-210m over 480k steps with gradient-colored layer log-curves (11 transformer layers). Let ϕ be the **SAE** extractor (Def. 27). Each dynamic uses distance $m_t = d(\phi(h_{\theta_t}), \phi(h_{\theta_T}))$. With distances d (left→right): Frobenius (FRO), Spectral (SPEC), Nuclear (NUCL), Entry-wise p (LP), Procrustes (PROC), Schatten p (S_P), Matrix Logarithm (M_LOG), Grassmannian (GRASS), Bures (BURES) [Sec. 8.4.1].

References

[AB99] Martin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 1999.

[ADRS+19] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, Raja Chatila, and Francisco Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai, 2019.

[BCB16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3(null):1137–1155, March 2003.

[BGBA+25] Nicolas Boizard, Hippolyte Gisserot-Boukhlef, Duarte M. Alves, André Martins, Ayoub Hammal, Caio Corro, Céline Hudelot, Emmanuel Malherbe, Etienne Malaboeuf, Fanny Jourdan, Gabriel Hautreux, João Alves, Kevin El-Haddad, Manuel Faysse, Maxime Peyrard, Nuno M. Guerreiro, Patrick Fernandes, Ricardo Rei, and Pierre Colombo. Eurobert: Scaling multilingual encoders for european languages, 2025.

[CCR24] Javier Cárcamo, Antonio Cuevas, and Luis A. Rodríguez. On uniqueness of the set of k-means, 2024.

[DCLT19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[DLJ10] Chris H.Q. Ding, Tao Li, and Michael I. Jordan. Convex and semi-nonnegative matrix factorizations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):45–55, 2010.

[DSB17] Derek Doran, Sarah Schulz, and Tarek R. Besold. What does explainable ai really mean? a new conceptualization of perspectives, 2017.

[FdRL17] Elena Facco, Maria d'Errico, Alessandro Rodriguez, and Alessandro Laio. Estimating the intrinsic dimension of datasets by a minimal neighborhood information. *Scientific Reports*, 7(1):12140, 2017.

[Goo15] Shlens J. Szegedy C. Goodfellow, I.J. Explaining and harnessing adversarial examples. *ICLR*, (2015).

[GVL13] Gene H. Golub and Charles F. Van Loan. *Matrix Computations - 4th Edition*. Johns Hopkins University Press, Philadelphia, PA, 2013.

[HO00] A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411–430, 2000.

[HP99] Aapo Hyvärinen and Petteri Pajunen. Nonlinear independent component analysis: Existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

[KWG+18] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav), 2018.

[LH19] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[LS99] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, 1999.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297, 1967.

[MBPS09] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 689–696, New York, NY, USA, 2009. Association for Computing Machinery.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[Ng11] Andrew Y. Ng. Sparse autoencoder. Course notes / technical report, Stanford University, 2011. CS294A / CS229 lecture notes (Sparse representation / unsupervised feature learning).

[OF96] Bruno A. Olshausen and David J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[OMLV+12] Sandra Ortega-Martorell, Paulo J. Lisboa, Alfredo Vellido, Rui V. Simões, Martí Pumarola, Marta Julià-Sapé, and Carles Arús. Convex non-negative matrix factorization for brain tumor delimitation from mrsi data. *PLoS ONE*, 7(10):e47824, 2012.

[PBE+22] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

[PCP+23] Eleonora Poeta, Gabriele Ciravegna, Eliana Pastor, Tania Cerquitelli, and Elena Baralis. Concept-based explainable artificial intelligence: A survey, 2023.

[PDV16] W. Pan and F. Doshi-Velez. A characterization of the non-uniqueness of nonnegative matrix factorizations, 2016.

[Pea01] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[POL+24] Core Francisco Park, Maya Okawa, Andrew Lee, Hidenori Tanaka, and Ekdeep Singh Lubana. Emergence of hidden capabilities: Exploring learning dynamics in concept space, 2024.

[PSF+25] Isabel Papadimitriou, Huangyuan Su, Thomas Fel, Sham Kakade, and Stephanie Gil. Interpreting the linear structure of vision-language model embedding spaces, 2025.

[PT10] Luca Pulina and Armando Tacchella. An abstraction-refinement approach to verification of artificial neural networks. volume 616, pages 243–257, 07 2010.

[RHW86] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[SDBD21] Samuel L Smith, Benoit Dherin, David Barrett, and Soham De. On the origin of implicit regularization in stochastic gradient descent. In *International Conference on Learning Representations*, 2021.

[SW24] Naomi Saphra and Sarah Wiegreffe. Mechanistic?, 2024.

[Vap98] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley-Interscience, 1998.

[VSP+23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

[YWS14] Yi Yu, Tengyao Wang, and Richard J. Samworth. A useful variant of the davis–kahan theorem for statisticians, 2014.

[ZBH+17] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.