Continual, On-the-Fly Learning through Sequential, Lightweight Optimization

Guy Lavi

Managing Partner

Vision Elements

Vision Elements

Powerhouse for *computational sciences* missions, requiring modeling, simulation, or optimization for *extraction of information* in computer vision, signal processing, deep learning, fluid dynamics, medical devices, augmented reality, robotics, and alike.

VisionElements

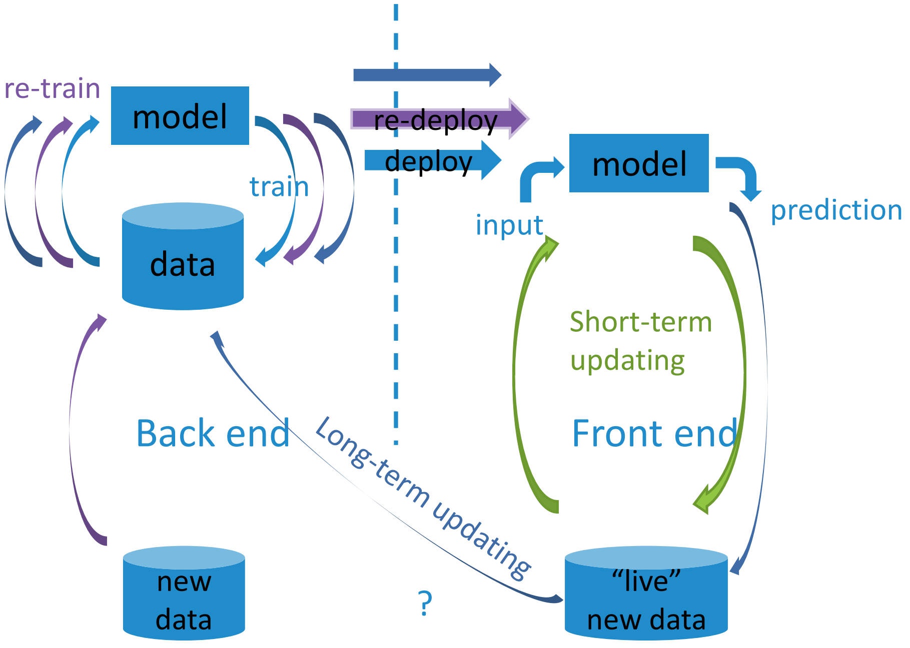
Continual, on-the-fly learning through sequential, lightweight optimization

Techniques of sequential optimization ➔ continual learning during run-time

The lightweight nature ➔ new training iterations on the edge

Without losing memory

Continual, on-the-fly learning through sequential, lightweight optimization



Paradigm of "living" model

Continual

* not replacing

Strategy for retraining (Replay)

Optimize sequentially using a memory term

Continual learning – state of the art

Continual learning – problem statement

Continual learning is a sub-field of machine learning, which aims to allow machine learning models to continuously learn on new data, by accumulating knowledge without forgetting what was learned in the past.

Static learning suffers from –

(1) inability to adapt to changing conditions,

(2) inability to improve with time,

(3) requires huge amounts of data collected and tagged apriori, and

(4) requires heavy resources for every cycle.

Continual learning therefore should allow for –

(1) adaptability to new data,

(2) improvement with experience,

(3) a faster deployment of solutions,

(4) lightweight resources that can live at the Edge, and utilization of user feedback as supervision.

As most AI solutions are static, many are sub-optimal or becoming outdated soon in their lifecycle.

Continual learning – terms and key concepts

Continual vs. Online Continual learning

Replay

i.i.d

CL STRATEGIES

Catastrophic forgetting

Continual learning – terms and key concepts

i.i.d (Independent and Identically Distributed) refers to a set of data points that are assumed to be drawn independently from the same probability distribution. Each data point is identically distributed, meaning that they share the same underlying statistical properties, such as when training models on a random sample of data.

Catastrophic Forgetting occurs when a model trained on multiple tasks forgets information about previously learned tasks as it learns new ones. This phenomenon can lead to a significant drop in performance on old tasks when adapting to new ones.

Replay is a technique used to mitigate catastrophic forgetting. Replay involves periodically revisiting old data (samples from previous tasks) during training; thus, the model can retain knowledge about earlier tasks while learning new ones.

Experience replay is a specific form of replay used in reinforcement learning. It involves storing past experiences (state-action-reward-next state tuples) in a replay buffer. During training, the agent samples from this buffer to learn from a mixture of recent and past experiences. Experience replay helps stabilize training and improve sample efficiency.

Online Continual Learning specifically focuses on learning from a continuous stream of data or tasks in an online fashion. Models adapt incrementally as new data arrives, without the need for explicit task boundaries or batch processing.

Strategy refers to an approach or methodology used to address the challenges posed by learning multiple tasks sequentially. Strategies can include architectural modifications, regularization techniques, memory-based methods, and replay mechanisms.

Continual learning – state of the art

ContinualAI
Lab

Avalanche is the flagship open-source collaborative project of **ContinualAI**: *a non profit research organization and the largest open community on Continual Learning for AI.*

6 papers

438 ★

Continual learning – state of the art

Major findings of our performance evaluation on Online Continual Learning

- Good stability does not necessarily transfer to higher accuracy (See Table 2, Figure 3 and Section 6 **Stability**).
- There is no best-performing OCL method across all metrics or memory sizes (See Table 2).
- OCL methods suffer from under-fitting in the common experimental setup (See Figure 3 and Section 6 **Forgetting**).
- Well properly tuned ER is a very competitive baseline obtaining better results than most existing methods (See **memory batch size** discussion 6 and Section 5 **Implementation**).
- The quality of the representation is very close to the one learned on the i.i.d stream, indicating that learning a good classifier is one of main problems. (See Section 6 **Representation quality**).

SCR [33]	Contrastive Loss, NMC	2021	
RAR [25]	Adversarial Augmentations	2022	
DER++ [5]	Distillation Loss	2020	
GDumb [38]	Offline finetuning on the buffer	2020	✓

Table 1: Summary of methods tried in the survey along with their particularities (release year, access to task boundaries).

arXiv:2308.10328v3

Joost van de Weijer
Computer Vision Center
Universitat Autònoma de Barcelona
Barcelona, Spain
joost@cvc.uab.cat

Abstract

Online continual learning aims to get closer to a live learning experience by learning directly on a stream of data with temporally shifting distribution and by storing a minimum amount of data from that stream. In this empirical evaluation, we evaluate various methods from the literature that tackle online continual learning. More specifically, we focus on the class-incremental setting in the context of image classification, where the learner must learn new classes incrementally from a stream of data. We compare these methods on the Split-CIFAR100 and Split-TinyImagenet benchmarks, and measure their average accuracy, forgetting, stability, and quality of the representations, to evaluate various aspects of the algorithm at the end but also during the whole training period. We find that most methods suffer from stability and underfitting issues. However, the learned representations are comparable to i.i.d. training under the same computational budget. No clear winner emerges from the results and basic experience replay, when properly tuned and implemented, is a very strong baseline. We release our modular and extensible codebase at https://github.com/AlbinSou/ocl_survey based on the avalanche framework to reproduce our results and encourage future research.

1. Introduction

In recent years, we have witnessed a surge of interest and progress in deep continual learning methodologies. These methods are able to learn continually from a stream of non-stationary data, thereby relaxing the principal assumption of having access to *independent and identically distributed* (*i.i.d.*) samples, often made in statistical learning [21]. In classic (or batch) continual learning, the common assumption is that the data stream is composed of distinct, explicitly defined *tasks* or *domains*, and that the method can detect the task boundaries or easily switch between domains. However, in many real-world scenarios, the data stream may not have clear task boundaries or domain labels, and the method

PDF
2308.10328.pdf

embedded VISION

VisionElements

State-of-the-art continual learning – key takeaways

Most continual learning strategies follow roughly the same training/evaluation loops, i.e., a simple naive strategy (a.k.a. finetuning) augmented with additional behavior to counteract catastrophic forgetting.

Overall, results are inferior to i.i.d.

A simple experience replay is outperforming more sophisticated strategies

VisionElements

Sequential optimization

Concept of sequential optimization

Sequential optimization / step-by-step estimation / differential adjustment / Kalman filtering, is a least squares method involving the derivation of expressions for the current estimate in terms of the previous estimate plus a correction term using the rules of matrix partitioning.

(Tobey 1930, Tienstra 1956, Schmid & Schmid 1965, Kalman 1960)

Chopping a large optimization problem into smaller accumulating ones

Updating a set of unknown parameters as new observations are available

The key principle guiding sequential optimization is that updating sequentially must yield the same final result as would have been obtained from a complete, simultaneous solution.

Least squares optimization formulation

unkowns $\bar{X} = X^0 + X$. $(u \times 1)$

observations $\bar{L} = L + V$. $(n \times 1)$

model $F(\bar{X}, \bar{L}) = 0$. $(r \times 1)$

$$n + u > r > u$$

$$F(\bar{X}, \bar{L}) = F(X^\circ, L) + \left.\frac{\partial F}{\partial \bar{X}}\right|_{X^\circ, L} X + \left.\frac{\partial F}{\partial \bar{L}}\right|_{X^\circ, L} V = 0$$

$$W + AX + BV = 0$$

misclosure vector

design matrices

$N = A^T P A$

$$\hat{X} = -(\overbrace{A^T P A}^{N})^{-1} A^T P W \,.$$

$$\hat{K} = P(A\hat{X} + W) \,.$$

$$\hat{V} = P^{-1}\hat{K} = A\hat{X} + W \,.$$

$$\hat{X} = -(A^T (B P^{-1} B^T)^{-1} A)^{-1} A^T (B P^{-1} B^T)^{-1} W \,.$$

$$\hat{K} = (B P^{-1} B^T)^{-1} (A\hat{X} + W) \,.$$

$$\hat{V} = -P^{-1} B^T \hat{K}$$

$$\Leftarrow \; A\hat{X} + B\hat{V} + W = 0 \; \begin{cases} X \rightarrow \hat{X} \\ V \rightarrow \hat{V} \\ \boxed{\hat{V}^T P \hat{V} = minimum} \\ (K \rightarrow \hat{K}) \end{cases}$$

VisionElements

Sequential optimization formulation

Similarly, the sequential formulation will look like this:

balance

$$\hat{K}_k = \big(\underbrace{P_k^{-1} + A_k N_{k-1}^{-1} A_k^T}_{(n \times n)^*}\big)^{-1} \left(A_k \hat{X}_{k-1} + W_k\right) .$$

$$\hat{X}_k = \hat{X}_{k-1} - N_{k-1}^{-1} A_k^T \hat{K}_k \quad .$$

$$\hat{V}_k = P_k^{-1} \hat{K}_k \quad .$$

$$N_k^{-1} = N_{k-1}^{-1} - N_{k-1}^{-1} A_k^T \left(P_k^{-1} + A_k N_{k-1}^{-1} A_k^T\right)^{-1} A_k N_{k-1}^{-1} \quad .$$

$$N_{k-1}^{-1} = \left(A_{k-1}^T P_{k-1} A_{k-1}\right)^{-1}$$

*Memory term

Note that we carry the same A, P, and W matrices from previous slide

Examples

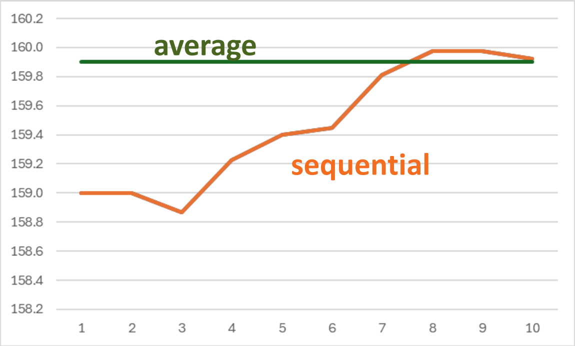
Sequential optimization in practice – application in linear optimization case

A beam of (true) length *l*:

$$\underset{(10 \times 1)}{L} = \begin{bmatrix} l_1 \\ l_2 \\ l_3 \\ \\ \vdots \\ l_{10} \end{bmatrix}$$ observations

F(X) = X – L = 0 model

$A = I$



direct

$$X = \bar{l} \pm \frac{\sigma_l}{\sqrt{10}}$$

average

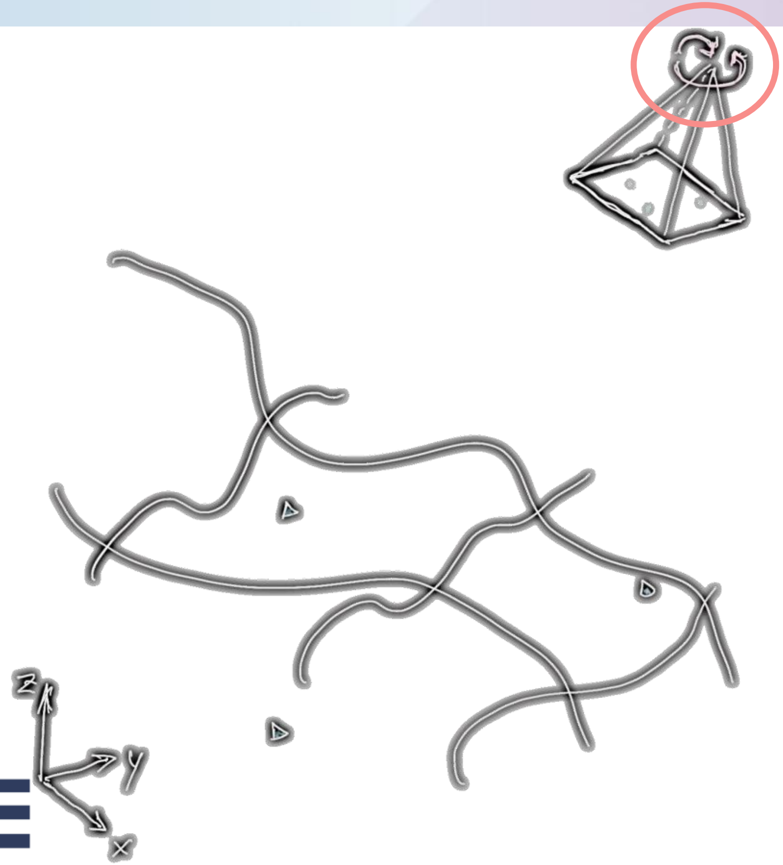
sequential

$$X_k = \bar{l}_k \pm \sigma_{\bar{l}_k}$$

updating average

$$X_{10} = \bar{l} \pm \frac{\sigma_l}{\sqrt{10}}$$

Sequential optimization – applicability to continual vision tasks



Single camera resection

unknowns

model

$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

observations

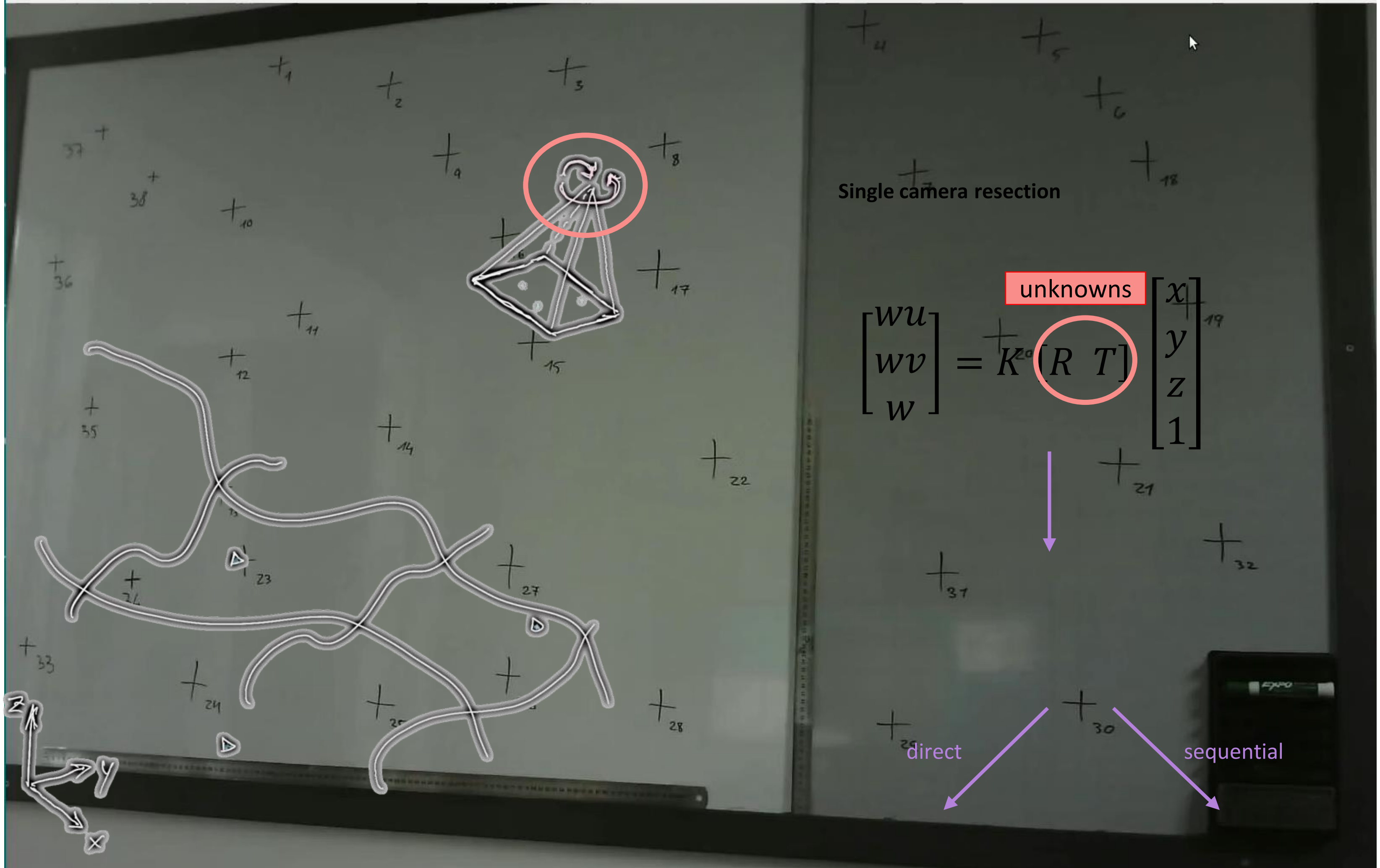
observations

$$r^{A}{}_{u} = \left.\frac{\partial F}{\partial \bar{X}}\right|_{X^{o},L} \qquad r^{B}{}_{n} = \left.\frac{\partial F}{\partial \bar{L}}\right|_{X^{o},L} \qquad r^{W}{}_{1} = F(X^{o}, L)$$

direct

sequential

VisionElements



Single camera resection

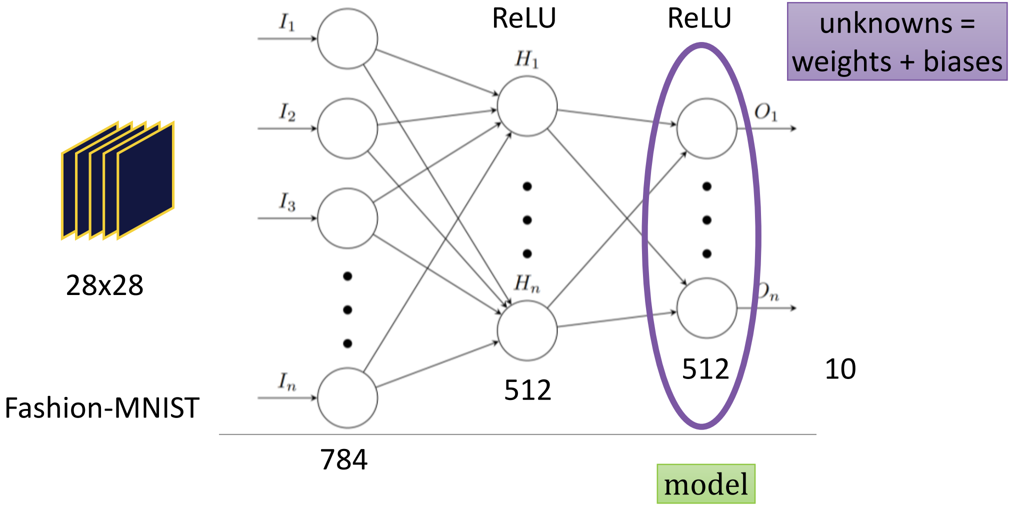
unknowns

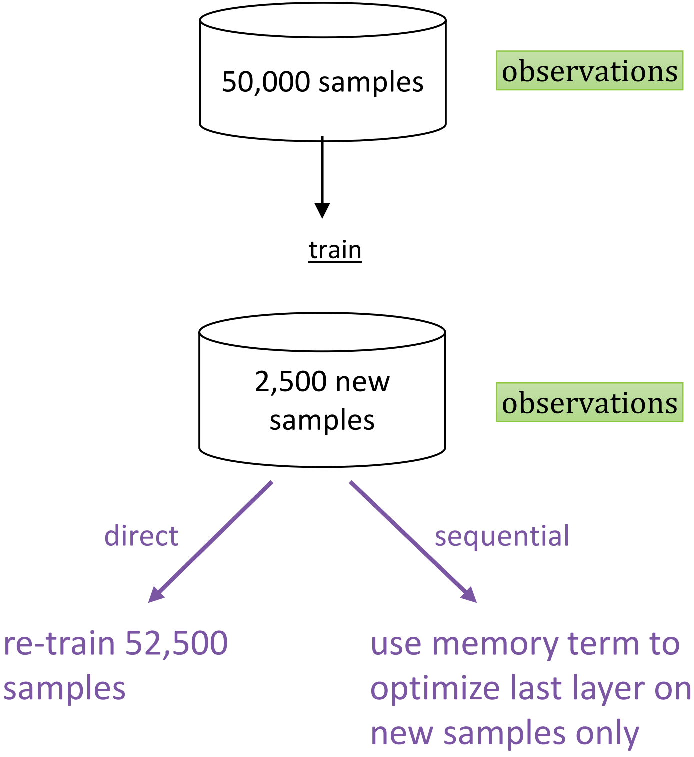
$$\begin{bmatrix} wu \\ wv \\ w \end{bmatrix} = K\begin{bmatrix} R & T \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

direct

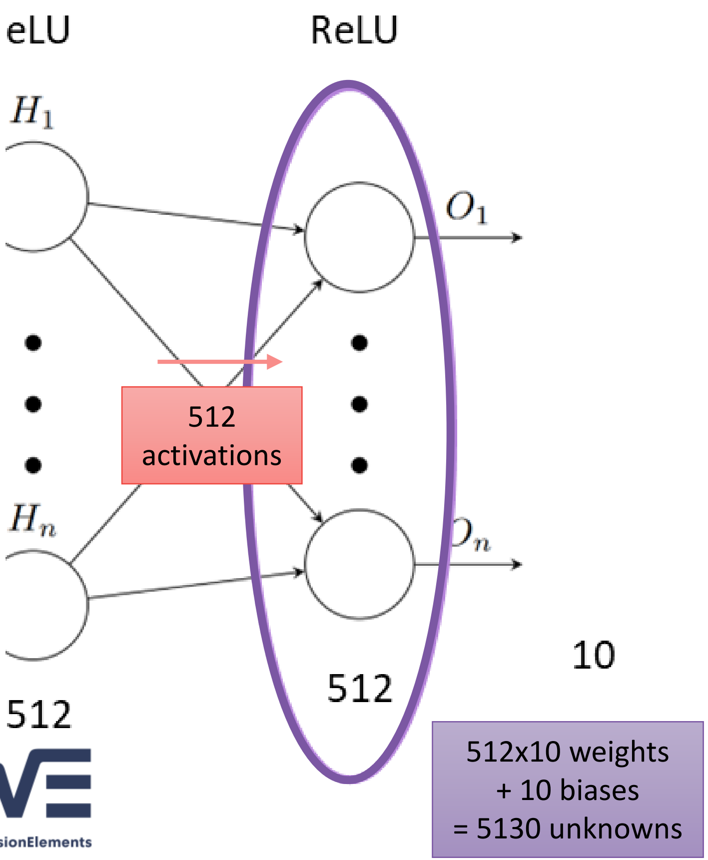
sequential

Application to object classification network

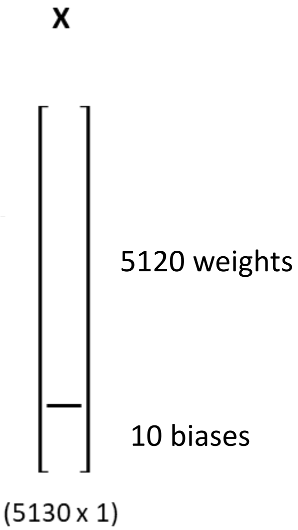




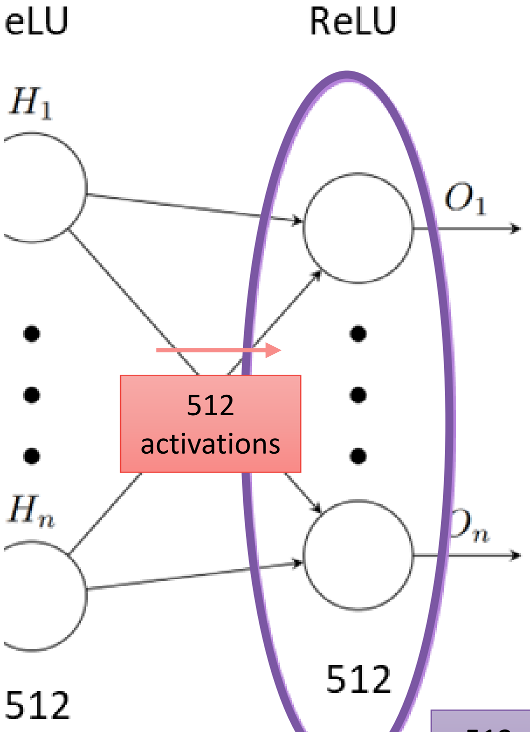
Application to object classification network



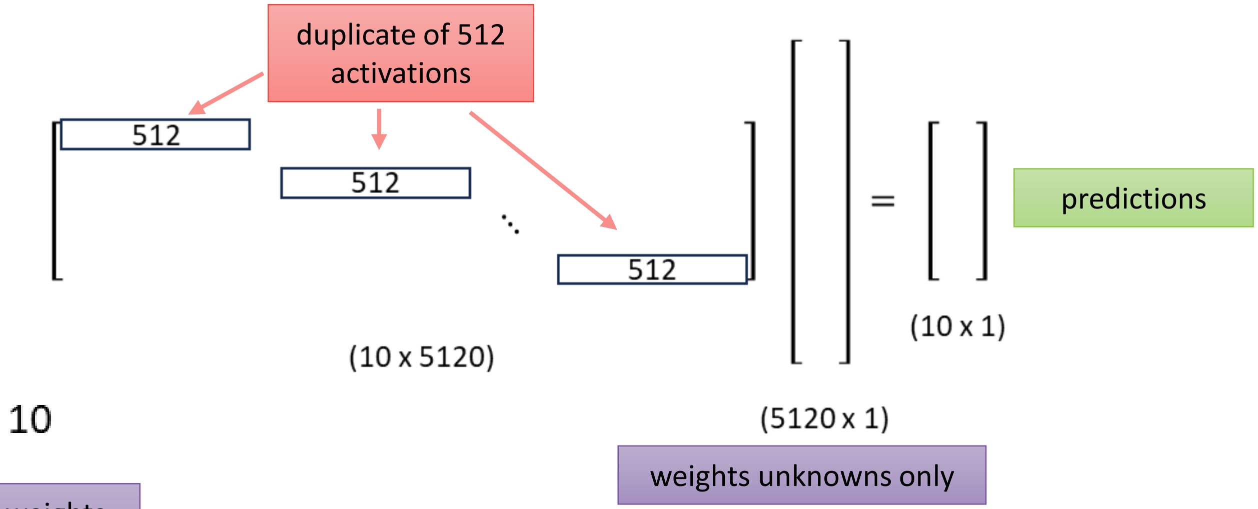
If we wanted to place all 5130 unknowns in one vector *X*, it would have looked like this:



Application to object classification network

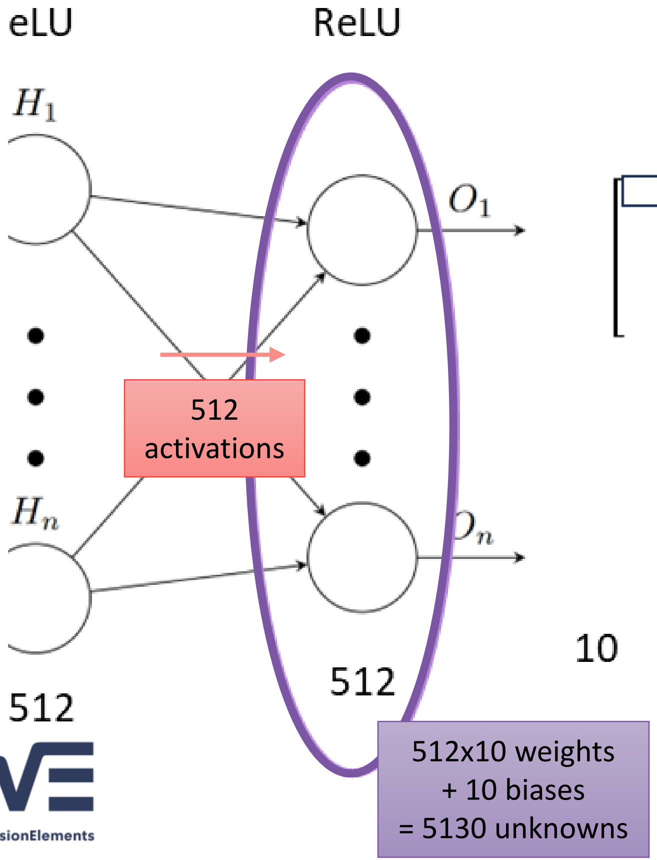


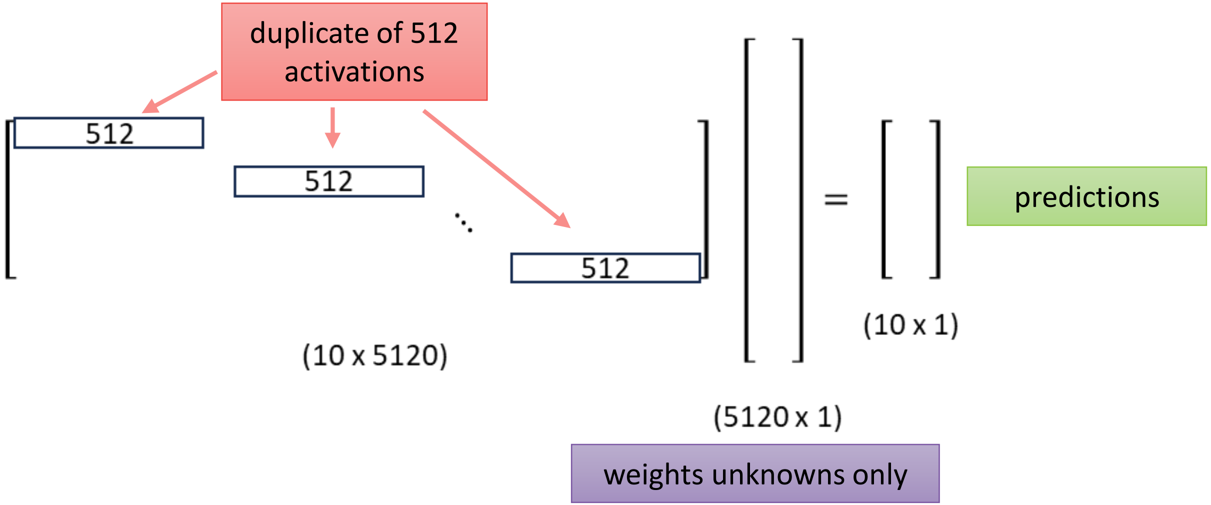
For simplicity, let's start with the weights only (so, X is 5120x1), and write their application to the incoming activations as matrix multiplication:



512x10 weights
+ 10 biases
= 5130 unknowns

Application to object classification network

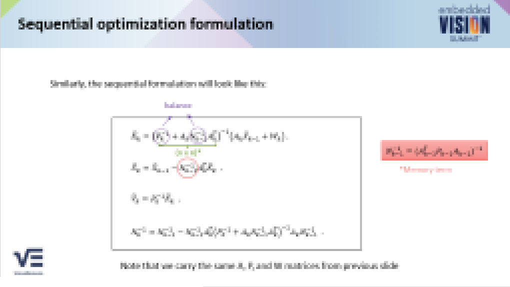




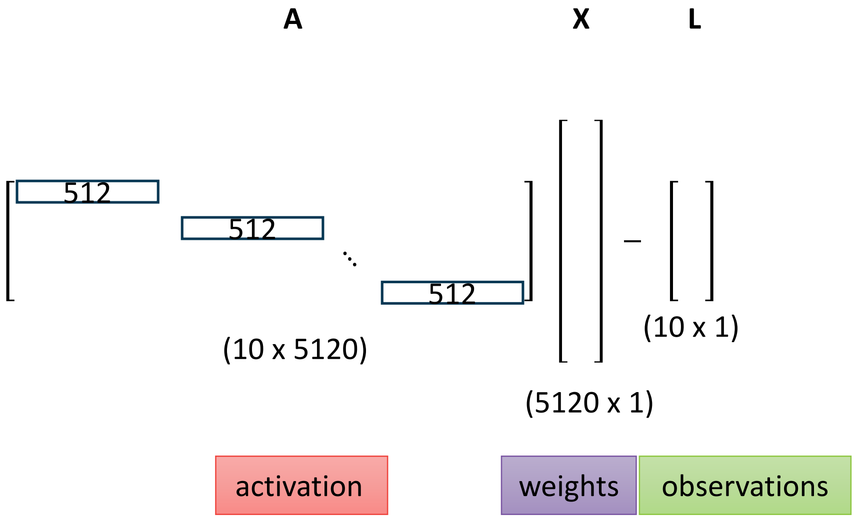
So, our model becomes *F(X, L) = AX – L = 0*, where *A* is the duplicate of activations matrix on the left, *X* are the unknowns, and *L* are the ground truth 'observations'.

model

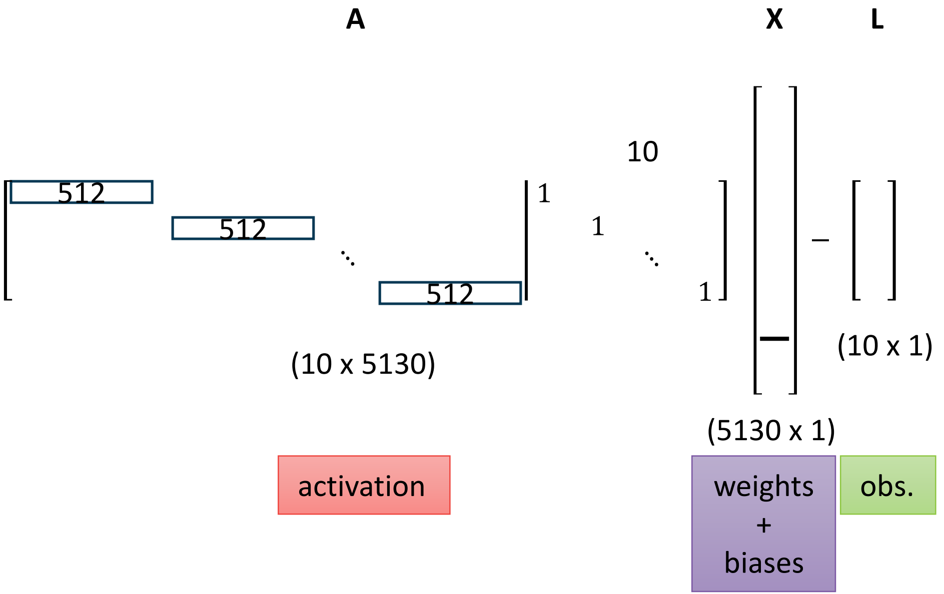
Application to object classification network



Let's write the last layer in matrix notations:



Adding biases, and rearranging *A*:

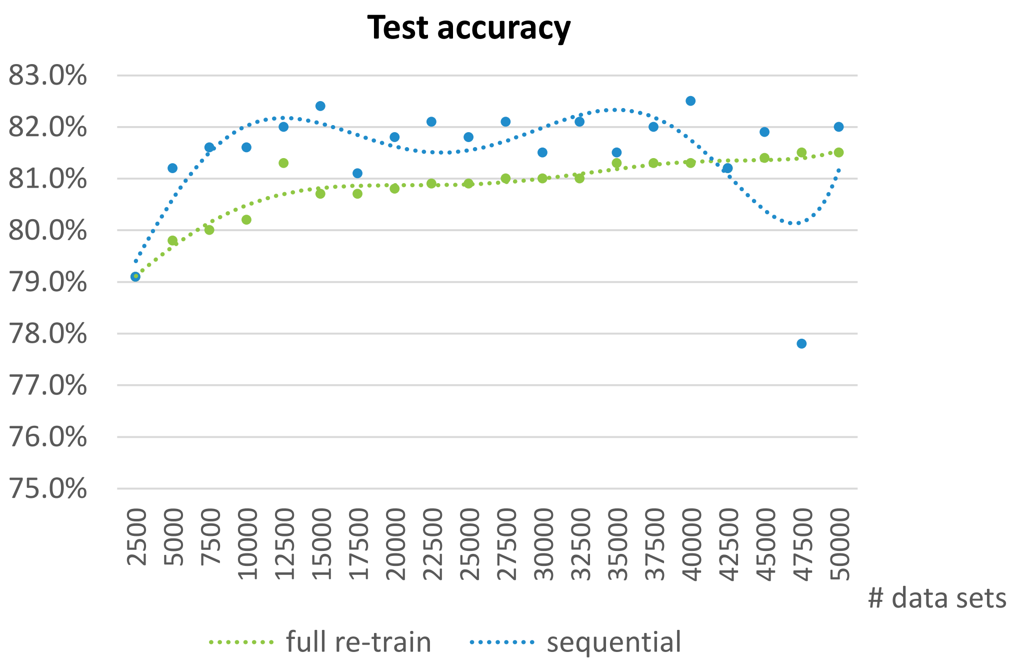


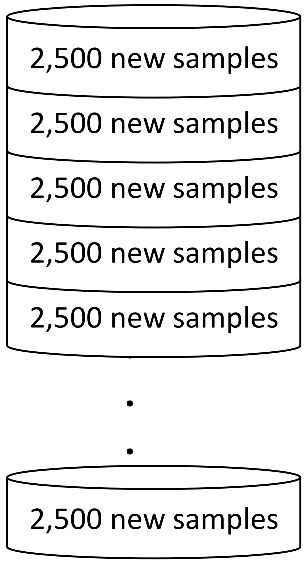
Application to object classification network

Database size	Accuracy	Steps
50000	88.2%	1000 epochs
+2500	88.3%	+1 step
52500	88.3%	1000 epochs

VisionElements

Application to object classification network





Sequential is ~ **x100** times faster

Summary and next steps

- A simple, 1-step, sequential optimization may obviate the need for heavy re-training
- Can potentially run on the Edge
- Learns new data without forgetting, utilizing a memory term
- Prone to drift, so does not replace re-training in the background
- Models can be deployed while data builds up
- Parameter space needs to be sliced for the optimization to focus on what matters. That is, as in the case of deeper neural networks, this method cannot accommodate the entire architecture at once.

- Future steps should involve generalizing to allow building systems that combine long-term, back-end with short-term, front-end updating

Resources

Avalanche

[Avalanche: an End-to-End Library for Continual Learning | Avalanche - v0.5.0 | Avalanche (continualai.org)]

Sequential optimization

Wells DE, Krakiwsky EJ, The Method of Least Squares, Lecture Notes 18, University of New Brunswick, May 1971

Continual, on-the-fly learning through sequential, lightweight optimization

[Detailed blog post]

Where to find us?

Guy Lavi, MSc
Managing Partner

Don't outsource. Deliver.

VisionElements

guy@vision-elements.com
www.vision-elements.com
+972 50 555 6040

AI & Scientists on Demand

VisionElements