Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών (Η.Μ.Μ.Υ.)



ΕΓΧΕΙΡΙΔΙΟ ΣΕΜΙΝΑΡΙΟΥ & ΕΡΓΑΣΤΗΡΙΑΚΟΣ ΟΔΗΓΟΣ

Τεχνητή Νοημοσύνη (AI), Μηχανική Μάθηση, Βαθιά Μάθηση & Νευρωνικά Δίκτυα

Από τα μαθηματικά θεμέλια του τεχνητού νευρώνα και τα συνελικτικά δίκτυα έως την αρχιτεκτονική Transformers, τα Μεγάλα Γλωσσικά Μοντέλα (LLMs), τα ανοιχτά μοντέλα (Open-Source, LoRA, RAG & AI Agents), τη Ραδιοαστρονομία SETI, τα 33 εφαρμοσμένα εργαστήρια σε Python 3.12+, την ηθική διακυβέρνηση και τους κβαντικούς υπολογιστές.

ΝΕΥΡΩΝΙΚΑ ΔΙΚΤΥΑ **LLMS & OPEN-SOURCE** **PYTHON 3.12 & SETI** **ΚΒΑΝΤΙΚΗ & ΗΘΙΚΗ**

18 Ενότητες
Πλήρης θεωρητική, μαθηματική και αρχιτεκτονική ανάλυση με μεγάλα, ευκρινή διαγράμματα.

33 Εργαστήρια Python
Σύγχρονοι, τεκμηριωμένοι κώδικες σε NumPy, Keras 3, PyTorch 2, LoRA, RAG, MCP & SETI (code.zip).

Επιστήμη & Άνθρωπος
Δημιουργική AI, «Αστόφως 1980», Ευρωπαϊκή Πράξη AI, κβαντική υπολογιστική και «Η Άδεια Θέση».

Ευάγγελος Β. Πεφάνης
Μέλος Ε.Τ.Ε.Π., Τμήμα Η.Μ.Μ.Υ., Πανεπιστήμιο Πελοποννήσου

Εκπαιδευτική Μονογραφία Σεμιναρίου
Πάτρα • Συνοδευτικός Κώδικας: code.zip

ΕΙΣΑΓΩΓΙΚΟ ΣΗΜΕΙΩΜΑ

Πρόλογος & Αφιέρωση

Το πνεύμα της έρευνας, οι ευχαριστίες και ο πρακτικός οδηγός μελέτης του σεμιναρίου.

Η παρούσα εργασία, η δημιουργία της παρουσίασης του σεμιναρίου και οι κώδικες των εξειδικευμένων εφαρμογών βασίζονται σε προσωπική μελέτη, έρευνα και συστηματική ενασχόληση με το αντικείμενο της Τεχνητής Νοημοσύνης, της Μηχανικής Μάθησης και της Βαθιάς Μηχανικής Μάθησης.

Όλη αυτή την προσπάθεια την αφιερώνω στο προσωπικό και στους φοιτητές του Πανεπιστημίου Πελοποννήσου που με τιμούν με τη συμμετοχή τους σε αυτό το σεμινάριο και ιδιαίτερα σε όλους εκείνους που αναζητούν την αλήθεια του κόσμου και των πραγμάτων μέσα από την έρευνα και κάθε δημιουργική έκφραση.

Η εργασία αυτή φιλοδοξεί να αποτελέσει ένα χρήσιμο και ουσιαστικό βοήθημα για τους φοιτητές και το ακαδημαϊκό προσωπικό, προσφέροντας ταυτόχρονα το έναυσμα για περαιτέρω ενασχόληση, πειραματισμό και επιστημονική έρευνα στον ραγδαία εξελισσόμενο αυτό τομέα.

Ευχαριστώ θερμά τον Πρόεδρο του Τμήματος Η.Μ.Μ.Υ., Καθηγητή κ. Σωτήριο Χριστοδούλου, που υποστήριξε τη διοργάνωση του σεμιναρίου και μου έδωσε τη δυνατότητα να προωθήσω αυτή μου την εργασία, καθώς επίσης και να εκφραστώ δημιουργικά υλοποιώντας το σύνολο του εκπαιδευτικού υλικού.

Ευάγγελος Β. Πεφάνης
Μέλος Ε.Τ.Ε.Π., Τμήμα Η.Μ.Μ.Υ., Πανεπιστήμιο Πελοποννήσου

Οδηγός Μελέτης & Αξιοποίησης του Συνοδευτικού Κώδικα (code.zip)

Το εγχειρίδιο είναι δομημένο σε δεκαοκτώ θεματικές ενότητες και έναν φιλοσοφικό επίλογο, ακολουθώντας πιστά τη λογική ροή της παρουσίασης και του εκπαιδευτικού βίντεο του σεμιναρίου. Κάθε θεωρητική ενότητα συνδέεται άμεσα με αντίστοιχα εκτελέσιμα εργαστήρια σε **Python 3.12+**, τα οποία περιλαμβάνονται στο συνοδευτικό αρχείο `code.zip`.

Θεωρία & Μαθηματικά

Κάθε αρχιτεκτονική (Perceptron, MLP, CNN, RNN, LSTM, Transformer, LLM, LoRA, RAG, DQN) αναλύεται με διαφανείς μαθηματικές εξισώσεις και διαγράμματα υψηλής ευκρίνειας.

33 Εργαστήρια Python

Τα αρχεία `01_*.py` έως `33_*.py` διαθέτουν ελληνικά σχόλια, σύγχρονα APIs (Keras 3, PyTorch 2, Ollama, LoRA, RAG, MCP & SETI) και επαληθευμένα αποτελέσματα.

Επιστήμη & Όραμα

Από την υπολογιστική επιτάχυνση GPU, τα ανοιχτά LLMs και τη δημιουργική AI έως την Ευρωπαϊκή Πράξη AI, τα λογικά qubits και την ανθρώπινη ευθύνη.

▸ **Συνιστώμενη Σειρά Εργαστηριακής Εξάσκησης**

Ξεκινήστε από τα αυτοτελή προγράμματα NumPy (`02_and_gate_numpy.py`, `06_xor_gate_numpy.py`, `22_transformer_self_attention.py`, `23_bpe_tokenizer_and_sampling.py` έως `29_ai_agent_tool_calling_react.py`) που εκτελούνται άμεσα σε οποιονδήποτε υπολογιστή χωρίς ειδικό υλικό, και προχωρήστε στα μοντέλα βαθιάς μάθησης (MNIST, CIFAR-10, IMDb, Καιρός Πάτρας, Τρίλιζα DQN και Δημιουργική AI) αξιοποιώντας υποστήριξη CUDA GPU όπου είναι διαθέσιμη.

ΔΟΜΗ ΜΟΝΟΓΡΑΦΙΑΣ

Πίνακας Περιεχομένων

Δεκαοκτώ θεματικές ενότητες, 33 εκτελέσιμα εργαστήρια Python και ένας ανθρώπινος επίλογος. Κάντε κλικ σε κάθε ενότητα για άμεση μετάβαση.

Ενότητα	Τίτλος Κεφαλαίου	Σελίδα
Πρόλογος	Πρόλογος, Αφιέρωση & Οδηγός Μελέτης	2
Κεφάλαιο 1	Θεμέλια Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης	4
Κεφάλαιο 2	Βιολογικοί και Τεχνητοί Νευρώνες, Συναρτήσεις Ενεργοποίησης	9
Κεφάλαιο 3	Λογικές Πύλες, Μαθηματική Βελτιστοποίηση & Πρώτα Μοντέλα	14
Κεφάλαιο 4	Βαθιές Αρχιτεκτονικές: MLP, CNN, RNN, LSTM και ConvLSTM2D	26
Κεφάλαιο 5	Αρχιτεκτονική Transformers και Μετωπικά Μοντέλα AI	32
Κεφάλαιο 6	Αξιολόγηση, Γενίκευση και Κύκλος Ζωής Μοντέλων	37
Κεφάλαιο 7	Οικοσύστημα Λογισμικού, Οπτικοποίηση & Επιθεώρηση Μοντέλων	43
Κεφάλαιο 8	Υπολογιστική Επιτάχυνση GPU, CUDA & Περιβάλλον Εργασίας	48
Κεφάλαιο 9	Εσωτερική Αρχιτεκτονική LLMs: BPE, RoPE, KV-Cache & Δειγματοληψία	54
Κεφάλαιο 10	Ανοιχτά Μοντέλα (Open-Source AI), Ollama & 4-Bit Κβαντοποίηση	61
Κεφάλαιο 11	Εξειδίκευση Μοντέλων, LoRA / QLoRA & Ευθυγράμμιση (Alignment)	67
Κεφάλαιο 12	Συστήματα RAG, Διανυσματικές Βάσεις & Αυτόνομοι Πράκτορες AI (MCP)	72
Κεφάλαιο 13	Εφαρμοσμένα Εργαστήρια Τεχνητής Νοημοσύνης (Όραση, Γλώσσα, Καιρός, RL)	78
Κεφάλαιο 14	Ραδιοαστρονομία SETI & Εκπαίδευση Astro-LLMs (Labs 30 έως 33)	89
Κεφάλαιο 15	Δημιουργική Νοημοσύνη: Μουσική, Καλειδοσκόπιο & «Αστόφως 1980»	106
Κεφάλαιο 16	Ηθική, Διακυβέρνηση και η Επιστημονική Επανάσταση της AI	111
Κεφάλαιο 17	Κβαντικοί Υπολογιστές, Κβαντική AI & Μετακβαντική Κρυπτογραφία	115
Κεφάλαιο 18	Ο Δρόμος προς την Εξειδίκευση: Μαθηματικά, Κώδικας & Έρευνα	120
Επίλογος	«Η Άδεια Θέση» & Ανθρωπότητα και Νοημοσύνη προς το Σύμπαν	123

Συνοπτικός Χάρτης Εργαστηρίων Python (code.zip)

Ενότητα Εργαστηρίων	Αρχεία Κώδικα	Βιβλιοθήκες & Τεχνολογίες
Νευρώνες & Λογικές Πύλες	`01` έως `08`	Tkinter, NumPy, h5py (H5Web), Keras 3, TensorBoard
Μετρικές & Επιτάχυνση GPU	`09` έως `12`	NumPy, Pandas, PyTorch 2 (CUDA/AMP), TensorFlow / Keras 3
LLMs, Open-Source, LoRA, RAG & Agents	`23` έως `29`	BPE, Mini-GPT, INT4 GGUF, Ollama, LoRA, Cosine RAG, ReAct/MCP
Εφαρμοσμένη & Δημιουργική AI	`13` έως `22`	CNN, Bi-LSTM, ConvLSTM2D, Open-Meteo, DQN, MIDI, Transformers
Ραδιοαστρονομία SETI & Astro-LLMs	`30` έως `33`	CSIRO simulateSearch, ABACAD, Spectral ViT, RTX 3060, LoRA & MCP Astro-Agent

Γρήγορη Εκκίνηση Περιβάλλοντος Python 3.12+: Αποσυμπιέστε το `code.zip` και εκτελέστε διαδοχικά: `python -m venv .venv` → `pip install -r requirements.txt` → `python 02_and_gate_numpy.py`



ΚΕΦΑΛΑΙΟ 1

Θεμέλια Τεχνητής Νοημοσύνης και Μηχανικής Μάθησης

Από την εξέλιξη των ανθρώπινων εργαλείων στην ιεραρχία AI, ML, Deep Learning, τα μαθηματικά παραδείγματα μάθησης και την Ελαχιστοποίηση Εμπειρικού Κινδύνου.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Διάκριση της ένθετης επιστημονικής ιεραρχίας **AI → Μηχανική Μάθηση (ML) → Βαθιά Μάθηση (DL) → Παραγωγική AI** και κατανόηση της μετάβασης από τα χειροποίητα χαρακτηριστικά στη Μάθηση Αναπαραστάσεων.
- Αυστηρή μαθηματική διατύπωση των τεσσάρων κυρίαρχων παραδειγμάτων μάθησης: **Επιβλεπόμενη**, **Μη Επιβλεπόμενη**, **Αυτοεπιβλεπόμενη** και **Ενισχυτική Μάθηση**.
- Κατανόηση των πέντε θεμελιωδών πυλώνων κάθε συστήματος ML, της αρχής της **Ελαχιστοποίησης Εμπειρικού Κινδύνου (ERM)** και του **Κενού Γενίκευσης (Generalization Gap)**.
- Αναγνώριση του ρόλου της **Επαγωγικής Προδιάθεσης (Inductive Bias)** στην επιλογή αρχιτεκτονικής ανάλογα με τη γεωμετρική δομή των δεδομένων.

1.1 Η Εξέλιξη της Ανθρώπινης Επινόησης: Από το Δόρυ στη Γνωσιακή Μηχανή

Σε ολόκληρη την ανθρώπινη ιστορία, η πρόοδος καθοδηγήθηκε από την κατασκευή εργαλείων που επέκτειναν τα βιολογικά μας όρια. Από τα πρώτα λίθινα εργαλεία της σαβάνας έως τη μηχανική κίνηση, τον ηλεκτρισμό, τους ψηφιακούς υπολογιστές και την εξερεύνηση του διαστήματος, κάθε τεχνολογικό άλμα πολλαπλασίασε τη μυϊκή ή υπολογιστική μας ικανότητα. Η **Τεχνητή Νοημοσύνη (Artificial Intelligence, AI)** συνιστά το επόμενο ποιοτικό άλμα: εργαλεία που δεν εκτελούν απλώς άκαμπτες προκαθορισμένες εντολές (*if-then-else*), αλλά εξάγουν στατιστικές κανονικότητες από δεδομένα, προσαρμόζονται σε νέα περιβάλλοντα και υποστηρίζουν σύνθετες αποφάσεις.



Σχήμα 1.1α

Η απαρχή της τεχνολογίας ως προέκταση της ανθρώπινης πρόθεσης και παρατήρησης.



Σχήμα 1.1β

Η συνέχεια της εξερεύνησης από τη Γη προς τα διαστημικά και υπολογιστικά σύνορα.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ **Πώς «Μαθαίνει» στην Πράξη ένας Υπολογιστής; Η Αναλογία της Συνταγής**

Αν ξεκινάτε σήμερα το ταξίδι σας στην Τεχνητή Νοημοσύνη, σκεφτείτε τη διαφορά ανάμεσα στο να δίνετε σε έναν φίλο μια *αυστηρή συνταγή* και στο να του μαθαίνετε να ξεχωρίζει ώριμα πορτοκάλια από την εμπειρία. Στον **κλασικό προγραμματισμό** γράφουμε ρητούς κανόνες (`if χρώμα == πορτοκαλί and διάμετρος > 7`). Στη **Μηχανική Μάθηση (ML)** αντιστρέφουμε τη διαδικασία: δίνουμε εκατοντάδες παραδείγματα με τη σωστή απάντηση και αφήνουμε τον αλγόριθμο να ανακαλύψει μόνος του τους μαθηματικούς κανόνες. Στη **Βαθιά Μάθηση (DL)** το δίκτυο προχωρά ένα βήμα παραπέρα, ανακαλύπτοντας αυτόματα ακόμη και ποια οπτικά στοιχεία (ακμές, υφή, σχήμα) αξίζει να προσέξει!

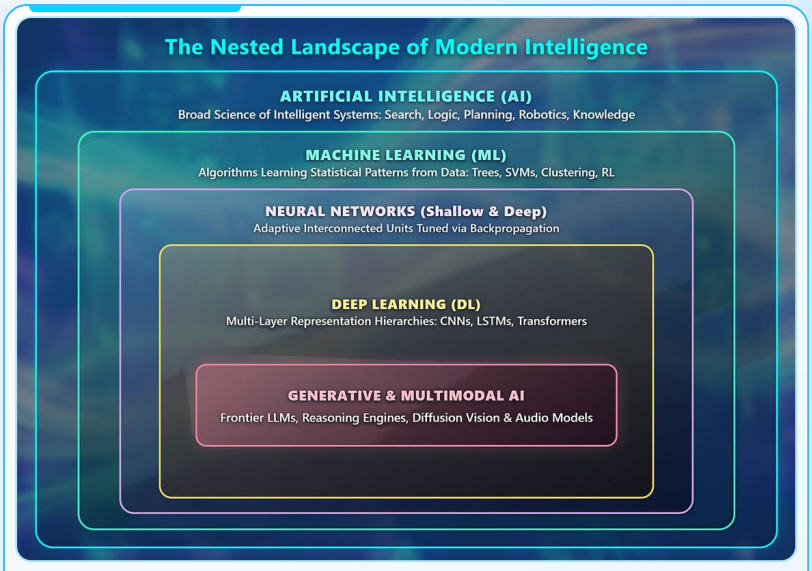
- **Χαρακτηριστικά (Features, *x*):** Οι αριθμητικές μετρήσεις εισόδου (π.χ. τετραγωνικά μέτρα, ώρες μελέτης, φωτεινότητα pixels).
- **Ετικέτα / Στόχος (Label, *y*):** Η πραγματική απάντηση που επιδιώκουμε να προβλέψει το μοντέλο.
- **Παράμετροι / Βάρη (*θ, w, b*):** Οι εσωτερικοί «διακόπτες» που ρυθμίζει αυτόματα ο υπολογιστής κατά την εκπαίδευση.
- **Συνάρτηση Απώλειας (Loss, *L*):** Το «βαθμολόγιο σφάλματος» που δείχνει πόσο απέχει η πρόβλεψη από την αλήθεια.
- **Γενίκευση (Generalization):** Η ικανότητα να προβλέπουμε σωστά σε *νέα, άγνωστα* παραδείγματα και όχι μόνο σε όσα αποστηθίσαμε.

1.2 Η Ιεραρχία: AI, ML, Βαθιά Μάθηση & Μάθηση Αναπαραστάσεων

Στον επιστημονικό λόγο οι όροι Τεχνητή Νοημοσύνη, Μηχανική Μάθηση και Βαθιά Μάθηση συχνά χρησιμοποιούνται εναλλακτικά, ωστόσο περιγράφουν αυστηρά ένθετα υποσύνολα με διακριτό ρόλο και υπολογιστικό βάθος:

- **Τεχνητή Νοημοσύνη (AI):** Το ευρύτερο επιστημονικό πεδίο σχεδίασης συστημάτων που εκτελούν έργα συνδεδεμένα με την ανθρώπινη νόηση (συλλογισμός, αντίληψη, γλώσσα, σχεδιασμός), συμπεριλαμβανομένων τόσο συμβολικών συστημάτων κανόνων όσο και στατιστικών μεθόδων.
- **Μηχανική Μάθηση (Machine Learning, ML):** Υποσύνολο της AI στο οποίο ένας αλγόριθμος βελτιώνει την απόδοσή του σε μια εργασία *T*, βάσει μετρικής *P*, μέσω εμπειρίας από δεδομένα *E*, χωρίς ρητή κωδικοποίηση των κανόνων από τον προγραμματιστή.
- **Βαθιά Μάθηση (Deep Learning, DL):** Εξειδίκευση της ML που αξιοποιεί πολυεπίπεδα τεχνητά νευρωνικά δίκτυα ικανά να εκτελούν αυτόματη **Μάθηση Αναπαραστάσεων (Representation Learning)** απευθείας από ακατέργαστα δεδομένα (εικόνες, ήχο, κείμενο, σήματα).
- **Παραγωγική AI (Generative AI):** Βαθιά θεμελιώδη μοντέλα (Transformers, LLMs, Diffusion) που εκτιμούν την υποκείμενη πιθανοτική κατανομή των δεδομένων ώστε να συνθέτουν πρωτότυπο κείμενο, κώδικα, εικόνες, μουσική και επιστημονικές υποθέσεις.

Για να αντιληφθούμε τη διαφορά μεταξύ **χειροποίητων χαρακτηριστικών (hand-crafted features)** και **μάθησης αναπαραστάσεων**, ας φανταστούμε πώς θα διδάσκαμε έναν υπολογιστή να αναγνωρίζει χειρόγραφους αριθμούς ή ιατρικές ακτινογραφίες. Στην κλασική ML ο μηχανικός σχεδίαζε χειροκίνητα φίλτρα ακμών, ιστογράμματα φωτεινότητας και γεωμετρικούς δείκτες, διαδικασία επίπονη και εύθραυστη. Αντίθετα, στη Βαθιά Μάθηση το δίκτυο συνθέτει μια *ιεραρχία εννοιών* από τα κάτω προς τα πάνω: τα πρώτα επίπεδα ανακαλύπτουν ακμές και αντιθέσεις, τα μεσαία συναρμολογούν υφές και γεωμετρικά μοτίβα και τα ανώτερα κωδικοποιούν ολόκληρες σημασιολογικές οντότητες, βελτιστοποιώντας ταυτόχρονα την εξαγωγή χαρακτηριστικών και την τελική απόφαση (*end-to-end learning*).



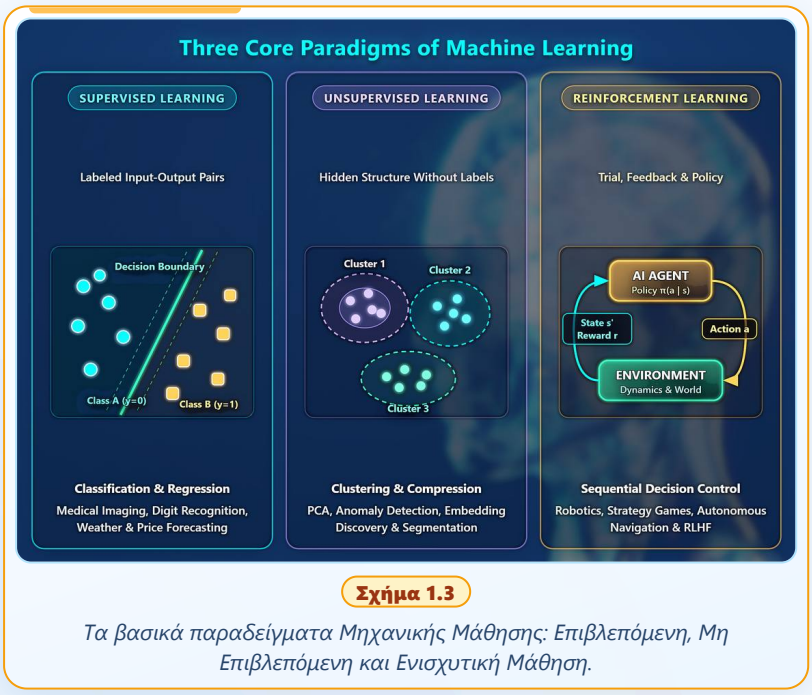
Σχήμα 1.2

Η ένθετη αρχιτεκτονική σχέση Τεχνητής Νοημοσύνης, Μηχανικής Μάθησης, Βαθιάς Μάθησης και Παραγωγικής AI.

1.3 Μαθηματική Διατύπωση των Παραδειγμάτων Μάθησης

Πριν περάσουμε στον συμβολισμό, αξίζει να «μεταφράσουμε» τρεις βασικές μαθηματικές έννοιες σε καθημερινή γλώσσα: ένα **διάνυσμα (vector)** είναι απλώς μια διατεταγμένη λίστα αριθμών (π.χ. τα χαρακτηριστικά ενός διαμερίσματος `[70 τ.μ., 2 υπνοδωμάτια, 3ος όροφος]` αποτελούν ένα διάνυσμα στον τρισδιάστατο χώρο R^3), ένας **πίνακας (matrix)** είναι ένας ορθογώνιος πίνακας αριθμών (όπως ένα φύλλο Excel ή μια ασπρόμαυρη εικόνα), ενώ μια **κατανομή πιθανότητας *P(X, Y)*** εκφράζει τους πραγματικούς στατιστικούς κανόνες του κόσμου που συνδέουν τις εισόδους με τα αποτελέσματα.

Η φύση των διαθέσιμων δεδομένων και του σήματος ανάδρασης καθορίζει το μαθηματικό παράδειγμα που επιλέγει ο μηχανικός. Κάθε μοντέλο ορίζει μια παραμετρική οικογένεια συναρτήσεων $f_\theta : X \to Y$ (τον **χώρο υποθέσεων**), όπου το διάνυσμα $\theta \in R^p$ συγκεντρώνει σε μία λίστα όλες τις p εκπαιδεύσιμες παραμέτρους του συστήματος:



Σχήμα 1.3

Τα βασικά παραδείγματα Μηχανικής Μάθησης: Επιβλεπόμενη, Μη Επιβλεπόμενη και Ενισχυτική Μάθηση.

Παράδειγμα Μάθησης	Μαθηματική Διατύπωση Δεδομένων & Στόχου	Κυρίαρχη Αντικειμενική Συνάρτηση	Χαρακτηριστικές Εφαρμογές
Επιβλεπόμενη (Supervised)	Ζεύγη $D = \{(x_i, y_i)\}_{i=1..N} \sim P(X, Y)$. Εκτίμηση $\hat{y} = f_\theta(x)$.	Ελαχιστοποίηση απώλειας $L(\hat{y}, y)$ (MSE για παλινδρόμηση, Cross-Entropy για ταξινόμηση)	Αναγνώριση ψηφίων MNIST, ιατρική διάγνωση, πρόγνωση καιρού Πάτρας
Μη Επιβλεπόμενη (Unsupervised)	Ανετικέτα δείγματα $D = \{x_i\}_{i=1..N}$. Εκμάθηση πυκνότητας $p_\theta(x)$ ή λανθάνοντος χώρου $z = E_\varphi(x)$.	Σφάλμα ανακατασκευής $\|x - \hat{x}\|^2$, μεγιστοποίηση πιθανοφάνειας ή ενδοσυσταδικής συνοχής	Ομαδοποίηση k-Means, μείωση διαστάσεων PCA, ανίχνευση ανωμαλιών SETI
Αυτοεπιβλεπόμενη (Self-Supervised)	Παραγωγή αυτόματων στόχων $(\tilde{x}_i, y_i^{auto})$ απευθείας από τεράστια ανετικέτα σώματα.	Πρόβλεψη επόμενου token $-\Sigma \ln P_\theta(w_t \mid w_{<t})$ ή καλυμμένων τμημάτων (MLM)	Προεκπαίδευση Μεγάλων Γλωσσικών Μοντέλων (GPT, Gemma, Llama, BERT)
Ενισχυτική (Reinforcement, RL)	Μαρκοβιανή Διαδικασία Απόφασης (S, A, P, R, γ) με πολιτική $\pi_\theta(a\|s)$.	Μεγιστοποίηση αναμενόμενης προεξοφλημένης ανταμοιβής $J(\pi) = E_\pi[\Sigma_t \gamma^t r_t]$	Παιχνίδια στρατηγικής (Τρίλιζα DQN, AlphaGo), ρομποτικός έλεγχος, RLHF

1.4 Ελαχιστοποίηση Εμπειρικού Κινδύνου (ERM), Κενό Γενίκευσης & Επαγωγική Προδιάθεση

Πώς συνδέεται μαθηματικά μια απλή λογική πύλη XOR με ένα σύγχρονο γλωσσικό μοντέλο εκατοντάδων δισεκατομμυρίων παραμέτρων; Στη βαθύτερη ουσία τους, και τα δύο υλοποιούν τους ίδιους **πέντε καθολικούς πυλώνες**: *Δεδομένα → Χώρος Υποθέσεων → Διαφορίσιμη Απώλεια → Βελτιστοποιητής Κλίσης → Γενίκευση*. Διαισθητικά, ένας φοιτητής που αποστηθίζει μηχανικά τις απαντήσεις παλαιών θεμάτων πετυχαίνει άριστα στο σύνολο εξάσκησης, αλλά αποτυγχάνει στις τελικές εξετάσεις. Αντίστοιχα, στόχος της Μηχανικής Μάθησης δεν είναι απλώς ο μηδενισμός του σφάλματος στα γνωστά δείγματα εκπαίδευσης (**Εμπειρικός Κίνδυνος** $R_{emp}(\theta)$), αλλά η ελαχιστοποίηση του **Αληθούς Κινδύνου** $R(\theta)$ πάνω στην άγνωστη κατανομή νέων, αθέατων παρατηρήσεων $P(X, Y)$.

$$\theta^* = \text{argmin}_\theta \left[(1/N) \Sigma_{i=1..N} L(f_\theta(x_i), y_i) + \lambda \cdot \Omega(\theta) \right] \quad | \quad \Delta_{gen}(\theta) = R(\theta) - R_{emp}(\theta) = E_{(x,y)\sim P}[L(f_\theta(x), y)] - R_{emp}(\theta)$$

Αριστερά: Κανονικοποιημένη Ελαχιστοποίηση Εμπειρικού Κινδύνου (ERM), όπου το λ·Ω(θ) τιμωρεί την υπερβολική πολυπλοκότητα (π.χ. L2 / Weight Decay). Δεξιά: Το Κενό Γενίκευσης Δ_{gen} μετρά την απόκλιση μεταξύ αθέατης και εκπαιδευτικής απόδοσης.

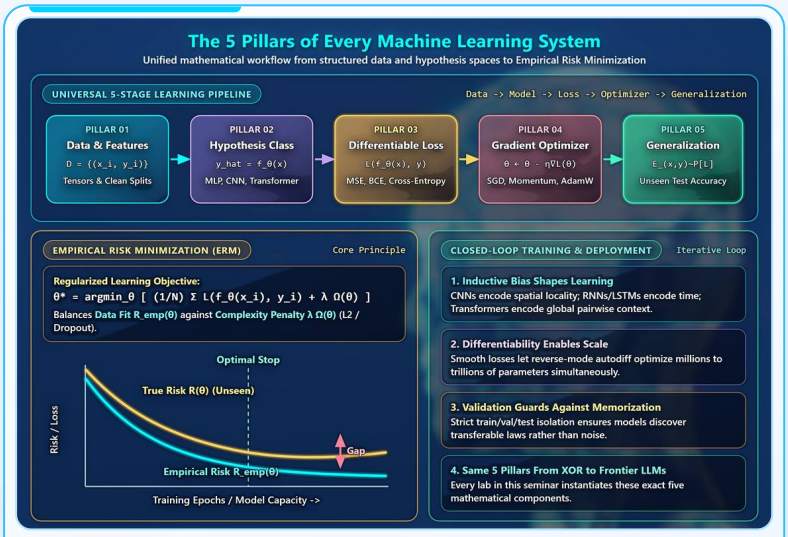
ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Πρόβλεψη Ενοικίου Διαμερίσματος: Κανόνες vs Μηχανική Μάθηση στο Χέρι

Έστω ότι διαθέτουμε 3 παραδείγματα διαμερισμάτων (εμβαδόν x σε τ.μ. → μηνιαίο ενοίκιο y σε ευρώ): `(50, 400)`, `(70, 520)` και `(90, 640)`. Στον **παραδοσιακό προγραμματισμό** θα γράφαμε εύθραυστους κανόνες `if/else` που αδυνατούν να προβλέψουν ενδιάμεσες τιμές. Στη **Μηχανική Μάθηση** επιλέγουμε τον απλό γραμμικό κανόνα $\hat{y} = w \cdot x + b$:

- **Βήμα 1 (Αρχική τυχαία υπόθεση $w = 4, b = 50$):** Οι προβλέψεις είναι $\hat{y}_1 = 4{\times}50 + 50 = 250$ (σφάλμα −150), $\hat{y}_2 = 4{\times}70 + 50 = 330$ (σφάλμα −190) και $\hat{y}_3 = 4{\times}90 + 50 = 410$ (σφάλμα −230).
- **Βήμα 2 (Μέτρηση Εμπειρικού Κινδύνου με MSE):** Υψώνουμε τα σφάλματα στο τετράγωνο ώστε να μην αλληλοαναιρούνται πρόσημα: $MSE = [(-150)^2 + (-190)^2 + (-230)^2] / 3 = (22.500 + 36.100 + 52.900) / 3 = 111.500 / 3 = 37.166{,}7$.
- **Βήμα 3 (Αυτόματη Βελτιστοποίηση & Γενίκευση):** Επειδή οι προβλέψεις είναι χαμηλές, η εκπαίδευση αυξάνει σταδιακά τα w, b έως τις βέλτιστες τιμές $w^* = 6$, $b^* = 100$ ($MSE = 0$). Τώρα για ένα **νέο, άγνωστο** διαμέρισμα 80 τ.μ. γενικεύει άμεσα: $\hat{y} = 6{\times}80 + 100 = 580$ €!
- **Βήμα 4 (Άλμα στη Βαθιά Μάθηση):** Αν αντί για έναν αριθμό (τ.μ.) δίναμε ακατέργαστες *φωτογραφίες* του σπιτιού, ένα βαθύ νευρωνικό δίκτυο (CNN) θα εξήγαγε μόνο του οπτικά χαρακτηριστικά (φωτεινότητα, ανακαίνιση, θέα) χωρίς χειροποίητους κανόνες.

Επειδή το πεπερασμένο δείγμα D δεν προσδιορίζει μονοσήμαντα τη συνάρτηση εκτός των σημείων εκπαίδευσης (Θεώρημα *No Free Lunch*), κάθε επιτυχημένο μοντέλο βασίζεται σε κατάλληλη **Επαγωγική Προδιάθεση (Inductive Bias)**: το σύνολο των δομικών υποθέσεων που ενσωματώνουμε στην αρχιτεκτονική του ώστε να γενικεύει αποτελεσματικά. Τα Συνελικτικά Δίκτυα (CNNs) ενσωματώνουν χωρική τοπικότητα και ισομεταβλητότητα σε μεταθέσεις, τα αναδρομικά δίκτυα (RNN/LSTM) ενσωματώνουν χρονική αιτιότητα, ενώ οι Transformers ενσωματώνουν καθολική συσχέτιση ανά ζεύγη μεταξύ όλων των στοιχείων της ακολουθίας.



Σχήμα 1.4

Οι πέντε θεμελιώδεις πυλώνες κάθε συστήματος Μηχανικής Μάθησης: ενοποιημένη ροή από τα δεδομένα και τον χώρο υποθέσεων στην Ελαχιστοποίηση Εμπειρικού Κινδύνου (ERM), το κενό γενίκευσης και την επαγωγική προδιάθεση.

◆ Κομβικά Ιστορικά Ορόσημα της Επιστήμης της AI

1956 (Συνέδριο Dartmouth): Γέννηση του όρου Τεχνητή Νοημοσύνη • **1986:** Ευρεία καθιέρωση της Οπισθοδιάδοσης Σφάλματος (Backpropagation) • **2012 (AlexNet):** Επανάσταση των βαθιών CNN με επιτάχυνση GPU • **2017 (Transformers):** Εισαγωγή της Αυτοπροσοχής • **Σύγχρονη Εποχή:** Πολυτροπικά θεμελιώδη μοντέλα, συλλογισμός και επιστημονικοί πράκτορες AI.

ΚΕΦΑΛΑΙΟ 2

Βιολογικοί και Τεχνητοί Νευρώνες, Συναρτήσεις Ενεργοποίησης

Από τη νευροβιολογική συναπτική ολοκλήρωση στη γεωμετρία υπερεπιπέδου του Perceptron, τις παραγώγους ενεργοποιήσεων και τα φαινόμενα Vanishing Gradients και Dying ReLUs.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Συσχέτιση της βιολογικής συναπτικής ολοκλήρωσης (διεγερτικά/ανασταλτικά δυναμικά και δυναμικό ενέργειας) με το μαθηματικό μοντέλο του τεχνητού νευρώνα.
- Γεωμετρική ερμηνεία του υπερεπιπέδου απόφασης $\mathbf{w}^T\mathbf{x} + \mathbf{b} = \mathbf{0}$ ως προς το κάθετο διάνυσμα βαρών *w* και τη μετατόπιση πόλωσης *b*.
- Πλήρης ανάλυση των συναρτήσεων ενεργοποίησης (**Sigmoid, Tanh, ReLU, Leaky ReLU, GELU, Softmax**), των αναλυτικών τους παραγώγων και των παθολογιών εκπαίδευσης (*Vanishing Gradients* & *Dying ReLUs*).
- Διαδραστική επαλήθευση κορεσμού και μη γραμμικής απόκρισης στο **Εργαστήριο 01** (`01_basic_neuron_gui.py`).

2.1 Βιολογική Συναπτική Ολοκλήρωση & Γεωμετρία Υπερεπιπέδου του Perceptron

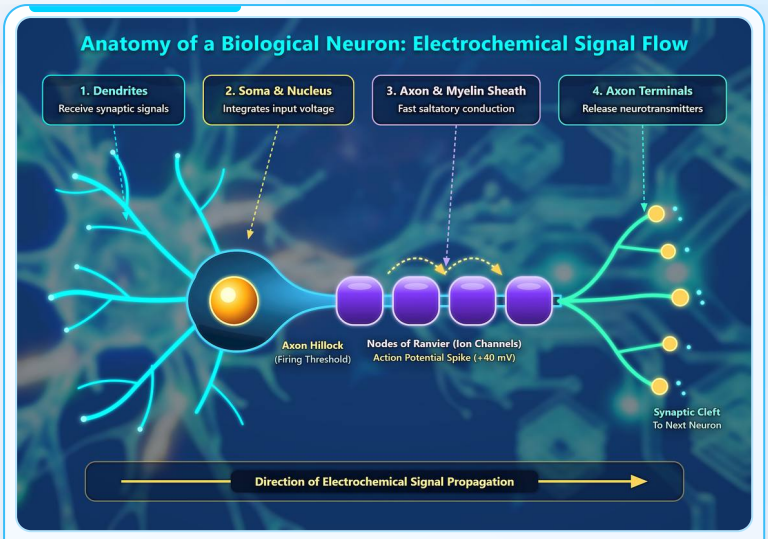
Ο ανθρώπινος εγκέφαλος περιέχει περίπου 86 δισεκατομμύρια νευρώνες συνδεδεμένους με εκατοντάδες τρισεκατομμύρια συνάψεις. Κάθε βιολογικός νευρώνας δέχεται μέσω των *δενδριτών* του χιλιάδες εισερχόμενες ηλεκτροχημικές ώσεις, οι οποίες διακρίνονται σε **διεγερτικές (EPSP)** που αυξάνουν το μεμβρανικό δυναμικό και **ανασταλτικές (IPSP)** που το μειώνουν. Οι ώσεις αυτές αθροίζονται χωρικά και χρονικά στο *κυτταρικό σώμα (soma)* και στον εκφυτικό κώνο του νευράξονα: εφόσον το συνολικό δυναμικό υπερβεί ένα κρίσιμο κατώφλι (περίπου −55 mV), ο νευρώνας πυροδοτεί ένα απότομο **δυναμικό ενέργειας (action potential)** «όλα ή τίποτα» κατά μήκος του μυελινωμένου *νευράξονα* προς τα επόμενα συναπτικά κομβία.

Ο τεχνητός νευρώνας (από το μοντέλο McCulloch-Pitts του 1943 και το Perceptron του Frank Rosenblatt του 1958 έως τις σύγχρονες μονάδες βαθιών δικτύων) μεταφράζει αυτή τη βιολογική αρχή σε δύο καθαρές μαθηματικές πράξεις: έναν **αφινικό γραμμικό συνδυασμό** των εισόδων *x* με τα εκπαιδεύσιμα βάρη *w* και τον όρο πόλωσης *b* (που μοντελοποιούν αντίστοιχα τη συναπτική ισχύ με θετικό/αρνητικό πρόσημο και το κατώφλι πυροδότησης), ακολουθούμενο από μία διαφορίσιμη **μη γραμμική συνάρτηση ενεργοποίησης** $f(z)$.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ **Ο Τεχνητός Νευρώνας ως Επιτροπή Αποφάσεων με Βάρη & Κατώφλι**

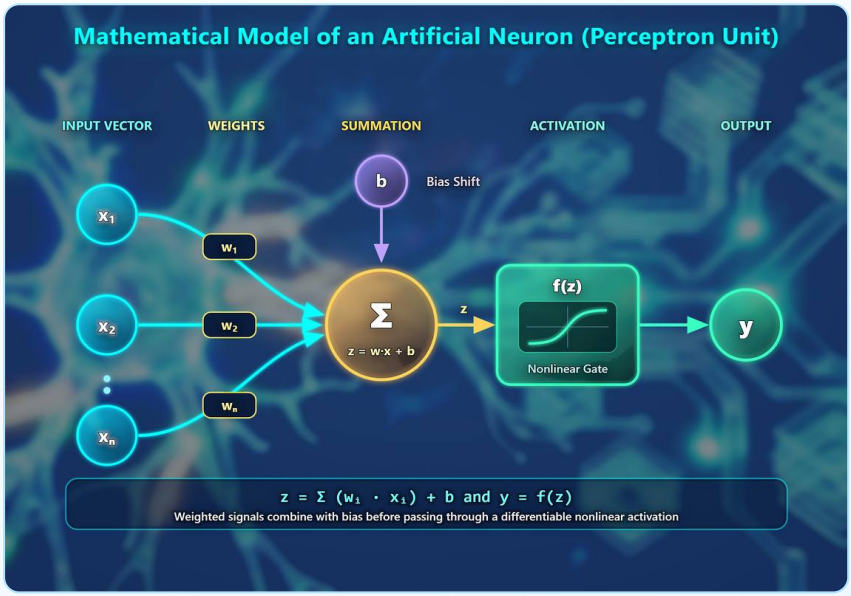
Φανταστείτε ότι αποφασίζετε αν θα πάτε στην παραλία με βάση τρία στοιχεία: θερμοκρασία (x_1), ηλιοφάνεια (x_2) και άνεμο (x_3). Κάθε παράγοντας έχει διαφορετική σημασία: η ηλιοφάνεια έχει θετικό **βάρος** (σπρώχνει προς το «Ναι»), ο δυνατός άνεμος έχει αρνητικό βάρος (σπρώχνει προς το «Όχι»), ενώ η **πόλωση (bias, *b*)** εκφράζει τη βασική σας διάθεση πριν καν κοιτάξετε τον καιρό. Αθροίζετε τις σταθμισμένες ψήφους σε μια συνολική βαθμολογία *z* και τη μετατρέπετε μέσω της **συνάρτησης ενεργοποίησης** $f(z)$ σε τελική απόφαση ή πιθανότητα!

- **Εσωτερικό Γινόμενο (w^Tx):** Απλώς η συντομογραφία του αθροίσματος $w_1x_1 + w_2x_2 + \dots + w_nx_n$.
- **Πόλωση (Bias, *b*):** Μετατοπίζει το κατώφλι ενεργοποίησης χωρίς να εξαρτάται από τις εισόδους.
- **Παράγωγος ($f'(z)$):** Η «κλίση» της καμπύλης στο σημείο *z*, δηλαδή πόσο ευαίσθητη είναι η έξοδος σε μικρή αλλαγή της εισόδου. Όταν $f'(z) \approx 0$ (**κορεσμός**), ο νευρώνας σταματά να μαθαίνει!
- **Logits:** Οι ακατέργαστες αριθμητικές βαθμολογίες *z* πριν μετατραπούν σε πιθανότητες (0% έως 100%) από Sigmoid ή Softmax.



Σχήμα 2.1α

Ανατομία βιολογικού νευρώνα (δενδρίτες, σώμα, νευράξονας, έλυτρο μυελίνης, συνάψεις).



Σχήμα 2.1β

Μαθηματικό διάγραμμα τεχνητού νευρώνα (Perceptron) με βάρη w, πόλωση b και ενεργοποίηση f(z).

$$\mathbf{z = w^T x + b = \Sigma_{i=1..n} (w_i \cdot x_i) + b \;\Rightarrow\; \hat{y} = f(z) \;\;|\;\; Υπερεπίπεδο\ Απόφασης:\ w^T x + b = 0 \;\; (d_0 = -b / ||w||_2)}$$

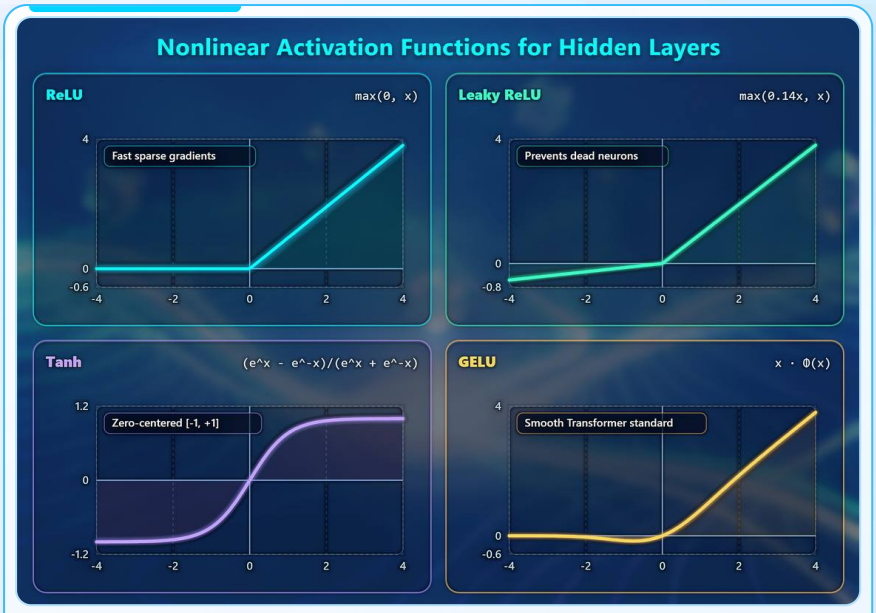
Το διάνυσμα βαρών w είναι κάθετο (normal vector) στο διαχωριστικό υπερεπίπεδο και καθορίζει τον προσανατολισμό και την απότομη κλίση της απόφασης, ενώ η πόλωση b μετατοπίζει το υπερεπίπεδο από την αρχή των αξόνων κατά απόσταση −b/||w||.

Η **γεωμετρική ερμηνεία** της εξίσωσης $w^T x + b = 0$ προσφέρει άμεση εποπτεία: σε δύο διαστάσεις ορίζει μια ευθεία, σε τρεις ένα επίπεδο και σε n διαστάσεις ένα **υπερεπίπεδο** που διχοτομεί τον χώρο εισόδου R^n σε δύο ημιχώρους ($z > 0$ και $z < 0$). Το διάνυσμα w δείχνει πάντα κάθετα προς την πλευρά της θετικής κλάσης: μεγαλύτερο μέτρο $||w||_2$ καθιστά τη μετάβαση της σιγμοειδούς αποτομότερη (μεγαλύτερη βεβαιότητα), ενώ χωρίς τον όρο πόλωσης b κάθε υπερεπίπεδο θα υποχρεωνόταν να διέρχεται από την αρχή των αξόνων $(0, ..., 0)$, αδυνατώντας να διαχωρίσει απλά μετατοπισμένα σύνολα σημείων.

2.2 Συναρτήσεις Ενεργοποίησης, Παράγωγοι, Vanishing Gradients & Dying ReLUs

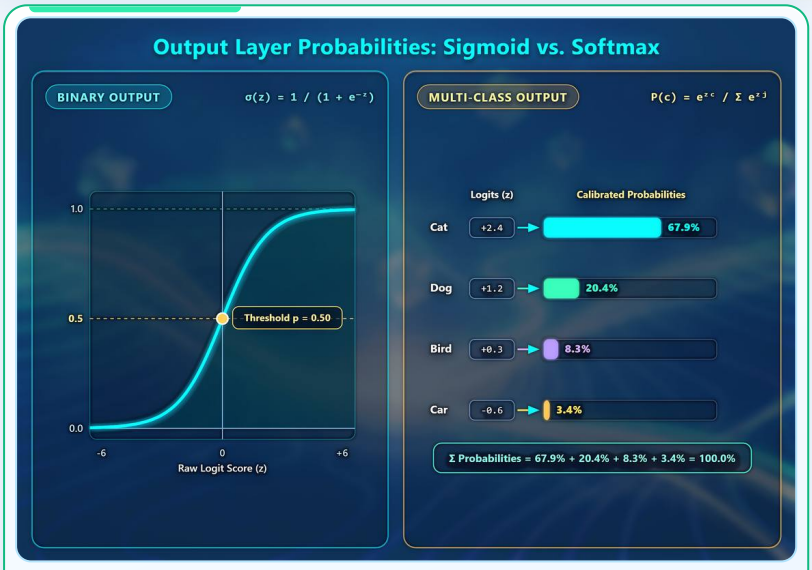
Χωρίς μη γραμμική συνάρτηση ενεργοποίησης, η σύνθεση οποιουδήποτε πλήθους γραμμικών επιπέδων καταρρέει αλγεβρικά σε έναν μοναδικό γραμμικό μετασχηματισμό: $W_2(W_1x + b_1) + b_2 = W_{eq}x + b_{eq}$. Ένα δίκτυο εκατό γραμμικών επιπέδων δεν θα είχε μεγαλύτερη εκφραστική ισχύ από έναν μόνο νευρώνα! Η μη γραμμικότητα επιτρέπει στα πολυεπίπεδα δίκτυα να «διπλώνουν» τον χώρο χαρακτηριστικών και να χαράσσουν αυθαίρετα καμπύλα όρια απόφασης. Παράλληλα, κατά την εκπαίδευση με οπισθοδιάδοση σφάλματος (Κεφάλαιο 3), οι κλίσεις πολλαπλασιάζονται διαδοχικά με την **παράγωγο f'(z)** κάθε επιπέδου, γεγονός που καθιστά τη μορφή της παραγώγου καθοριστική για τη σύγκλιση βαθιών αρχιτεκτονικών:

Συνάρτηση	Τύπος f(z) & Εύρος	Αναλυτική Παράγωγος f'(z)	Δυναμική Κλίσης & Μηχανικές Ιδιότητες
Sigmoid	$\sigma(z) = 1 / (1 + e^{-z})$ Εύρος: (0, 1)	$\sigma'(z) = \sigma(z) \cdot (1 - \sigma(z))$ Μέγιστο: 0.25 στο z = 0	**Εξασθένηση Κλίσης (Vanishing Gradients):** 10 επίπεδα μειώνουν την κλίση κατά $0.25^{10} \approx 10^{-6}$. Μη μηδενικά κεντραρισμένη. Χρήση κυρίως σε έξοδο δυαδικής πιθανότητας και πύλες LSTM.
Tanh	$(e^{z} - e^{-z}) / (e^{z} + e^{-z})$ Εύρος: (−1, +1)	$\tanh'(z) = 1 - \tanh^2(z)$ Μέγιστο: 1.0 στο z = 0	Μηδενικά κεντραρισμένη (ταχύτερη σύγκλιση από τη Sigmoid), αλλά οδηγείται επίσης σε κορεσμό ($f'(z) \to 0$) για $\lvert z\rvert > 3$. Ιδανική για εσωτερικές καταστάσεις RNN/LSTM.
ReLU	$\max(0, z)$ Εύρος: [0, +∞)	1 αν z > 0 0 αν z < 0	Σταθερή μοναδιαία κλίση στον θετικό ημιάξονα (λύση στα vanishing gradients) και ελάχιστο υπολογιστικό κόστος. Κίνδυνος **Νεκρών Νευρώνων (Dying ReLUs)** αν μεγάλο αρνητικό βήμα παγιδεύσει τον νευρώνα στο z < 0 με μόνιμη παράγωγο 0.
Leaky ReLU	$\max(\alpha z, z)$, $\alpha \approx 0.01$ Εύρος: (−∞, +∞)	1 αν z > 0 α αν z < 0	Εξαλείφει τους «νεκρούς νευρώνες» διατηρώντας μικρή μη μηδενική κλίση α στον αρνητικό ημιάξονα, επιτρέποντας την ανάκαμψη βαρών κατά την εκπαίδευση.
GELU	$0.5 \cdot z \cdot (1 + \text{erf}(z / \sqrt{2}))$ Εύρος: (−0.17, +∞)	$\Phi(z) + z \cdot \varphi(z)$ (Κανονική CDF + PDF)	Ομαλή, παντού διαφορίσιμη καμπύλη γύρω από το μηδέν με ελαφριά αρνητική βύθιση που δρα ως στοχαστική κανονικοποίηση. Πρότυπο σε Transformers και LLMs (BERT, GPT, Gemma).
Softmax	$e^{z_i} / \Sigma_j e^{z_j}$ Εύρος: (0, 1), Σ = 1	$\partial s_i/\partial z_j = s_i(\delta_{ij} - s_j)$ (Πλήρης Ιακωβιανός)	Μετατρέπει διάνυσμα ακανονικοποίητων *logits* σε έγκυρη κατανομή πιθανότητας *K* κλάσεων. Συνδυάζεται με *Cross-Entropy* προσφέροντας εξαιρετικά καθαρή κλίση $s - y$.



Σχήμα 2.2α

Κύριες μη γραμμικές συναρτήσεις κρυφών επιπέδων: ReLU, Leaky ReLU, Tanh και GELU.



Σχήμα 2.2β

Συναρτήσεις επιπέδου εξόδου: Sigmoid (δυαδική πιθανότητα) και Softmax (πολυκλαδική κατανομή).

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Υπολογισμός Τεχνητού Νευρώνα 2 Εισόδων στο Χέρι & Σύγκριση Ενεργοποιήσεων

Ας υπολογίσουμε έναν νευρώνα με εισόδους `x1 = 1.0, x2 = 0.5`, βάρη `w1 = 0.8, w2 = -0.4` και πόλωση `b = 0.2`:

- **Βήμα 1 (Γραμμικό δυναμικό z):** $z = w_1x_1 + w_2x_2 + b = (0.8 \times 1.0) + (-0.4 \times 0.5) + 0.2 = 0.80 - 0.20 + 0.20 = \mathbf{0.80}$.
- **Βήμα 2 (Σύγκριση ενεργοποιήσεων στο $z = 0.80$):** **ReLU(0.80)** = $max(0, 0.80)$ = **0.800** (παράγωγος 1.000) • **Sigmoid(0.80)** = $1 / (1 + e^{-0.80}) = 1 / 1.4493 = \mathbf{0.690}$ (69,0% πιθανότητα, παράγωγος $0.690 \times 0.310 = 0.214$) • **Tanh(0.80)** = $(2.2255 - 0.4493) / (2.2255 + 0.4493) = \mathbf{0.664}$ (παράγωγος $1 - 0.664^2 = 0.559$).
- **Βήμα 3 (Τι συμβαίνει σε ισχυρά αρνητικό δυναμικό $z = -5.0$;):** Η **ReLU(−5.0) = 0.000** με παράγωγο **0.000** (αν ο νευρώνας παγιδευτεί μόνιμα σε $z < 0$, παθαίνει *Dying ReLU*, ενώ η Leaky ReLU με $\alpha = 0.01$ κρατά τιμή -0.050 και ζωντανή κλίση 0.01). Η **Sigmoid(−5.0) = 0.0067** έχει παράγωγο μόλις **0.0066** και η **Tanh(−5.0) = −0.9999** έχει παράγωγο **0.00018** (σχεδόν μηδενικό σήμα μάθησης, αιτία των *Vanishing Gradients*!).

Κανόνας για αρχάριους: Στα εσωτερικά επίπεδα επιλέγουμε `ReLU` ή `GELU` ώστε οι κλίσεις να ρέουν ανεμπόδιστα, ενώ τη `Sigmoid` (ή `Softmax`) τη φυλάμε για το τελικό επίπεδο εξόδου.

2.3 Εργαστήριο 01: Διαδραστικός Προσομοιωτής Νευρώνα σε Python

ΕΡΓΑΣΤΗΡΙΟ 01 **Διαδραστικός Προσομοιωτής Τεχνητού Νευρώνα & Συναρτήσεων Ενεργοποίησης**

`code/01_basic_neuron_gui.py`

Το πρώτο εργαστήριο υλοποιεί ένα γραφικό περιβάλλον **Tkinter** με αριθμητικά ευσταθείς συναρτήσεις **Sigmoid** (με διακλάδωση πρόσημου για αποφυγή υπερχείλισης της `exp`), **ReLU**, **Tanh** και **GELU** (μέσω `math.erf`). Ο φοιτητής τροποποιεί σε πραγματικό χρόνο τις εισόδους x_i, τα βάρη w_i και την πόλωση b, παρακολουθώντας ζωντανά το γραμμικό δυναμικό $z = w^Tx + b$ και τη μη γραμμική απόκριση $y = f(z)$.

Προτεινόμενος Πειραματισμός: Δοκιμάστε ακραίες τιμές $z = +10$ και $z = -10$: διαπιστώστε πώς η **Sigmoid** και η **Tanh** κορέννυνται (σχεδόν μηδενική κλίση, πηγή *Vanishing Gradients*), πώς η **ReLU** σβήνει πλήρως για αρνητικά z (*Dying ReLU*), ενώ η **GELU** διατηρεί ομαλή, διαφορίσιμη μετάβαση κοντά στο μηδέν. Στη συνέχεια μεταβάλετε μόνο το b για να παρατηρήσετε τη μετατόπιση του κατωφλίου ενεργοποίησης.

PYTHON 3.12+ `code/01_basic_neuron_gui.py` — Πυρήνας αριθμητικά ευσταθών ενεργοποιήσεων & Tkinter GUI

```python
import math
import tkinter as tk
from tkinter import messagebox, ttk


def activate(z: float, func_name: str) -> float:
    """Εφαρμόζει την επιλεγμένη μη γραμμική συνάρτηση ενεργοποίησης στο z."""
    if func_name == "Sigmoid":
        # Αριθμητικά ευσταθής υλοποίηση της σιγμοειδούς 1 / (1 + e^(-z))
        # που αποτρέπει υπερχείλιση (overflow) της exp για μεγάλα αρνητικά z.
        if z >= 0:
            return 1.0 / (1.0 + math.exp(-z))
        exp_z = math.exp(z)
        return exp_z / (1.0 + exp_z)

    if func_name == "ReLU":
        # Rectified Linear Unit: αποκόπτει τις αρνητικές τιμές στο μηδέν
        # και διατηρεί γραμμική απόκριση για z > 0 χωρίς κορεσμό.
        return max(0.0, z)

    if func_name == "Tanh":
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/01_basic_neuron_gui.py του code.zip)
```

ΚΕΦΑΛΑΙΟ 3

Λογικές Πύλες, Μαθηματική Βελτιστοποίηση & Πρώτα Μοντέλα

Πώς μαθαίνει ένα νευρωνικό δίκτυο από το μηδέν: Πύλη AND, Κανόνας Αλυσίδας (Backpropagation), εξισώσεις Adam & AdamW, Functional API και το μη γραμμικό πρόβλημα XOR.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Κατανόηση του γραμμικού διαχωρισμού στη λογική πύλη **AND** και πλήρης υλοποίηση εμπρόσθιας και οπίσθιας διάδοσης σε καθαρή **NumPy** (**Εργαστήριο 02**).
- Αναλυτική παραγωγή του **Κανόνα της Αλυσίδας (Backpropagation)** πάνω σε υπολογιστικό γράφο και κατανόηση της απλοποίησης *BCE + Sigmoid*.
- Πλήρης μαθηματική διατύπωση των βελτιστοποιητών **Adam** και **AdamW** (ροπές m_t, v_t, διόρθωση μεροληψίας και αποσυνδεδεμένη φθορά βαρών).
- Σύγκριση **Keras 3 Sequential API** έναντι **Functional API** με παρακαμπτήριες συνδέσεις (*Skip Connections*, **Εργαστήρια 03 & 05**).
- Επίλυση του μη γραμμικού προβλήματος **XOR** με βάση το **Θεώρημα Καθολικής Προσέγγισης** και διαχωρισμός εκπαίδευσης από εξαγωγή συμπερασμάτων (**Εργαστήρια 06, 07 & 08**).

3.1 Η Λογική Πύλη AND ως Γραμμικά Διαχωρίσιμο Πρόβλημα

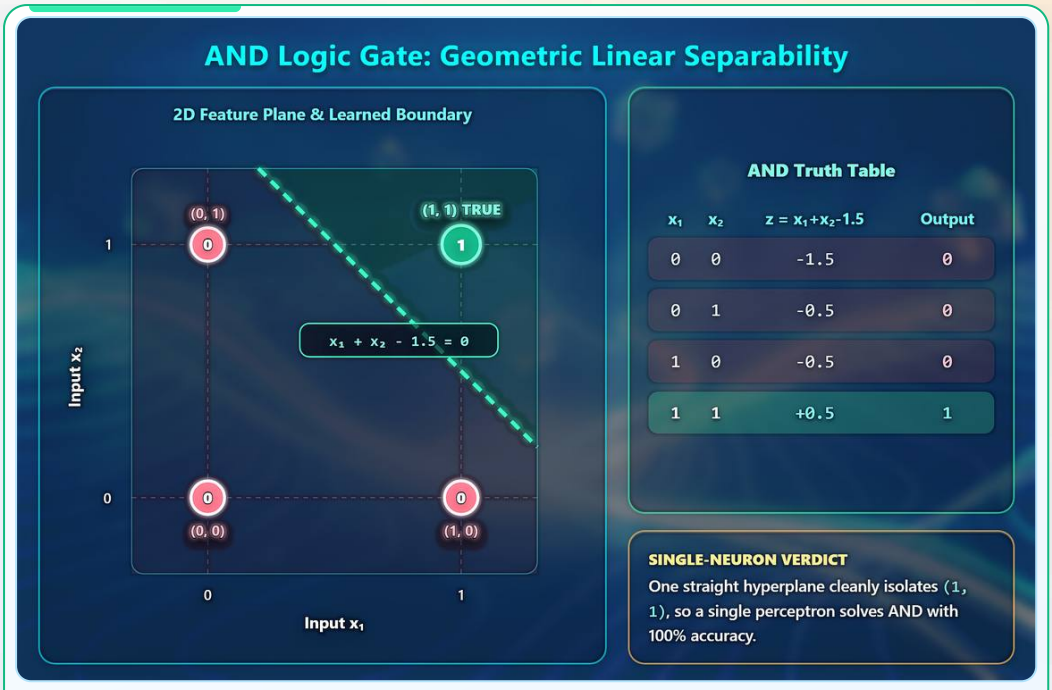
Η λογική πύλη **AND** αποτελεί το καθαρότερο εκπαιδευτικό πρότυπο στη μηχανική μάθηση: διαθέτει μόλις τέσσερα ζεύγη εισόδων `(0,0), (0,1), (1,0), (1,1)` και παράγει έξοδο `1` μόνο όταν και οι δύο είσοδοι είναι `1`. Γεωμετρικά, μια απλή ευθεία γραμμή στο δισδιάστατο επίπεδο (π.χ. $x_1 + x_2 - 1.5 = 0$) αρκεί για να απομονώσει την κορυφή `(1,1)` από τα υπόλοιπα τρία σημεία.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Κάθοδος Κλίσης (Gradient Descent): Κατεβαίνοντας ένα Βουνό μέσα στην Ομίχλη

Φανταστείτε ότι βρίσκεστε στην πλαγιά ενός βουνού μέσα σε πυκνή ομίχλη και θέλετε να φτάσετε στο χαμηλότερο σημείο της κοιλάδας (ελάχιστο σφάλμα). Δεν βλέπετε ολόκληρο τον χάρτη, αλλά νιώθετε την **κλίση (παράγωγο)** κάτω από τα πόδια σας: αν το έδαφος ανηφορίζει δεξιά (*θετική κλίση*), κάνετε ένα βήμα αριστερά· αν κατηφορίζει δεξιά (*αρνητική κλίση*), κάνετε ένα βήμα δεξιά! Το μέγεθος της δρασκελιάς σας είναι ο **Ρυθμός Μάθησης (η)**: πολύ μικρά βήματα αργούν απελπιστικά, ενώ τεράστια άλματα σας πετούν στην απέναντι πλαγιά! Η **Οπισθοδιάδοση Σφάλματος (Backpropagation)** είναι ο μαθηματικός «απολογισμός ευθύνης» που υπολογίζει αυτήν την κλίση για κάθε βάρος του δικτύου.

- **Εποχή (Epoch):** Ένα πλήρες πέρασμα *όλων* των δειγμάτων εκπαίδευσης από το μοντέλο.
- **Μικρο-δέσμη (Mini-Batch) & Επανάληψη (Iteration):** Αντί να περιμένουμε όλα τα δεδομένα, ενημερώνουμε τα βάρη ανά μικρές ομάδες (π.χ. 32 δείγματα = 1 iteration). Για 320 δείγματα και batch 32, μία εποχή περιέχει 10 iterations.
- **Κλίση (Gradient, ∇L):** Διάνυσμα παραγώγων που δείχνει την κατεύθυνση μέγιστης *ανόδου* του σφάλματος (άρα αφαιρούμε $-\eta \nabla L$).
- **Ορμή (Momentum, AdamW):** Όπως μια βαριά σφαίρα που κυλά με αδράνεια, εξομαλύνει τις ταλαντώσεις και ξεπερνά επίπεδα σημεία.



Σχήμα 3.1

Πίνακας αληθείας, γεωμετρικό όριο απόφασης και νευρωνική υλοποίηση της λογικής πύλης AND.

ΕΡΓΑΣΤΗΡΙΟ 02 **Εκπαίδευση Πύλης AND σε Καθαρή NumPy, Αρχικοποίηση Xavier & Εξαγωγή HDF5**

`code/02_and_gate_numpy.py`

Κατασκευάζουμε ένα δίκτυο δύο επιπέδων (`2 → 3 → 1`) χωρίς καμία βιβλιοθήκη βαθιάς μάθησης, ώστε να αποκαλύπτεται κάθε πολλαπλασιασμός πινάκων της εμπρόσθιας διέλευσης και του Backpropagation. Χρησιμοποιούμε **αρχικοποίηση Xavier/Glorot** (διακύμανση $2 / (fan_{in} + fan_{out})$) ώστε οι ενεργοποιήσεις να ξεκινούν στη γραμμική περιοχή της σιγμοειδούς, και εξάγουμε τα εκπαιδευμένα βάρη σε αρχείο **HDF5 (.h5)** για επιθεώρηση στο H5Web.

Προτεινόμενος Πειραματισμός: Τροποποιήστε τον ρυθμό μάθησης (π.χ. `0.01` έναντι `2.0`) και παρατηρήστε την ταχύτητα μείωσης του RMSE στις 10.000 εποχές, καθώς και τα τελικά βάρη μέσα στο παραγόμενο αρχείο `.h5` .

PYTHON 3.12+ `code/02_and_gate_numpy.py` Εμπρόσθια διάδοση, Backpropagation & εξαγωγή HDF5 σε NumPy

```python
from pathlib import Path
import numpy as np

try:
    import h5py
except ImportError:
    h5py = None


def sigmoid(z: np.ndarray) -> np.ndarray:
    """Σιγμοειδής συνάρτηση ενεργοποίησης: απεικονίζει κάθε πραγματική τιμή στο (0, 1)."""
    # Περιορισμός στο [-500, 500] ώστε η εκθετική συνάρτηση να παραμένει αριθμητικά ευσταθής
    return 1.0 / (1.0 + np.exp(-np.clip(z, -500.0, 500.0)))


def sigmoid_derivative_from_output(a: np.ndarray) -> np.ndarray:
    """Υπολογίζει την παράγωγο της σιγμοειδούς ως προς την ήδη υπολογισμένη έξοδο a."""
    return a * (1.0 - a)


def main() -> None:
    """Εκπαιδεύει δίκτυο NumPy στη λογική πύλη AND και εξάγει τα βάρη σε HDF5."""
    # === 1. Προετοιμασία Δεδομένων (Πίνακας Αληθείας Πύλης AND) ===
    # Σταθερός σπόρος τυχαιότητας (seed) για πλήρη επαναληψιμότητα των πειραμάτων
    rng = np.random.default_rng(seed=42)
```

```
    # Δυαδικές είσοδοι X διαστάσεων (4, 2) και επιθυμητοί στόχοι y διαστάσεων (4, 1)
    X = np.array(
        [
            [0.0, 0.0],
            [0.0, 1.0],
            [1.0, 0.0],
            [1.0, 1.0],
        ],
        dtype=np.float64,
    )
    y = np.array([[0.0], [0.0], [0.0], [1.0]], dtype=np.float64)

    # === 2. Αρχικοποίηση Βαρών (Xavier/Glorot) & Υπερπαραμέτρων ===
    # Η διακύμανση 2 / (fan_in + fan_out) διατηρεί ισορροπημένο πλάτος ενεργοποιήσεων και κλίσεων
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/02_and_gate_numpy.py του code.zip)
```

ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
Εποχή 1 | RMSE: 0.491903   ->   Εποχή 10000 | RMSE: 0.000640
(0,0)->0.000003 [0]  |  (0,1)->0.000461 [0]  |  (1,0)->0.000421 [0]  |  (1,1)->0.998882 [1]
```

3.2 Συνάρτηση Απώλειας, Κανόνας της Αλυσίδας (Backpropagation) & Βελτιστοποιητές Adam / AdamW

Η εκπαίδευση ενός νευρωνικού δικτύου διατυπώνεται ως πρόβλημα συνεχούς **μαθηματικής βελτιστοποίησης**: αναζητούμε τα βάρη θ που ελαχιστοποιούν μια διαφορίσιμη συνάρτηση απώλειας $L(\theta)$ (δηλαδή το «ύψος» του σφάλματος για κάθε επιλογή βαρών). Η μερική παράγωγος $\partial L/\partial w_i$ απαντά στο απλό ερώτημα: *«αν αυξήσω ελάχιστα το βάρος w_i, θα μεγαλώσει ή θα μικρύνει το σφάλμα;»*. Για δυαδική ταξινόμηση, η μεγιστοποίηση της λογαριθμικής πιθανοφάνειας Bernoulli ισοδυναμεί ακριβώς με την ελαχιστοποίηση της **Δυαδικής Διασταυρούμενης Εντροπίας (Binary Cross-Entropy, BCE)**, η οποία επιβάλλει αυστηρή λογαριθμική ποινή ($-\ln(p) \to +\infty$) όταν το μοντέλο εκφράζει υψηλή βεβαιότητα για λάθος κλάση.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Κάθοδος Κλίσης στο Χέρι στην Παραβολή $L(w) = (w - 3)^2$ & Επίδραση του Learning Rate η

Έστω η απλούστερη συνάρτηση απώλειας $L(w) = (w - 3)^2$ (το αληθινό ελάχιστο είναι στο $w^* = 3$ με $L = 0$). Η παράγωγος (κλίση) είναι $g(w) = dL/dw = 2(w - 3)$ και ο κανόνας ενημέρωσης $w_{t+1} = w_t - \eta \cdot g(w_t)$. Ξεκινάμε από $w_0 = 0.0$ (αρχικό σφάλμα $L(0) = 9.00$, κλίση $g_0 = 2(0 - 3) = -6.0$):

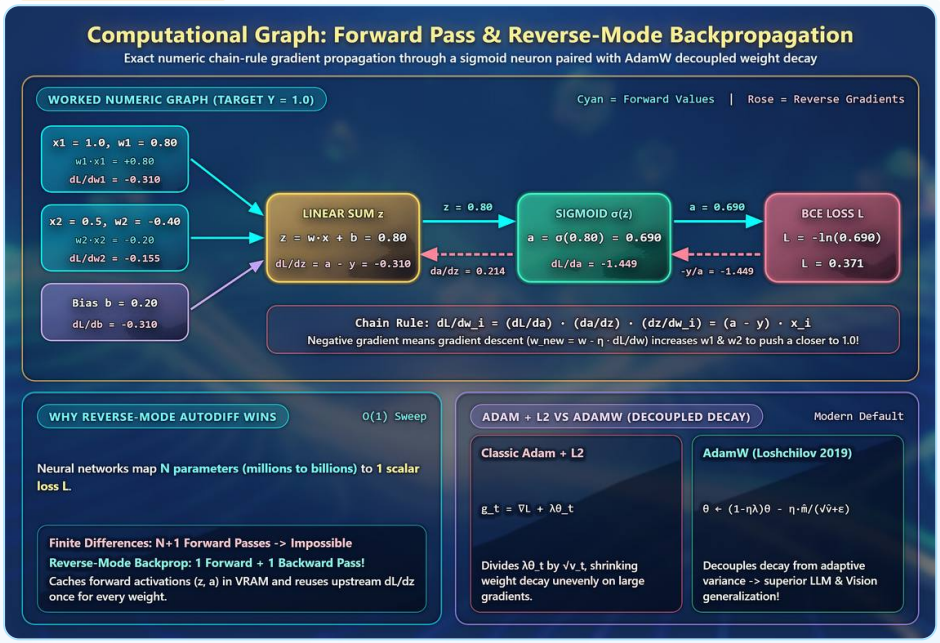
- **Ισορροπημένο $\eta = 0.1$ (ομαλή σύγκλιση):** Βήμα 1: $w_1 = 0.0 - 0.1(-6.0) = \mathbf{0.60}$ ($L = 5.76$). Βήμα 2: $g_1 = 2(0.60 - 3) = -4.8 \to w_2 = 0.60 - 0.1(-4.8) = \mathbf{1.08}$ ($L = 3.69$). Βήμα 3: $g_2 = -3.84 \to w_3 = \mathbf{1.464}$ ($L = 2.36$). Κάθε βήμα μειώνει την απόσταση από το 3 κατά 20%!
- **Υπερβολικά μικρό $\eta = 0.001$ (αργή πρόοδος):** Βήμα 1: $w_1 = 0.0 - 0.001(-6.0) = \mathbf{0.006}$ ($L = 8.964$). Χρειάζονται πάνω από 2.300 βήματα για να πλησιάσει το 3.0!
- **Υπερβολικά μεγάλο $\eta = 1.2$ (απόκλιση & έκρηξη!):** Βήμα 1: $w_1 = 0.0 - 1.2(-6.0) = \mathbf{+7.20}$ (υπερπήδησε το 3, $L = 17.64$!). Βήμα 2: $g_1 = +8.40 \to w_2 = 7.20 - 1.2(8.40) = \mathbf{-2.88}$ ($L = 34.57$!). Το σφάλμα εκτινάσσεται ανεξέλεγκτα (αιτία εμφάνισης `NaN` στην εκπαίδευση).

Πώς υπολογίζει όμως ένα δίκτυο με εκατομμύρια ή δισεκατομμύρια βάρη πόσο ευθύνεται *κάθε μεμονωμένο βάρος* w_i για το τελικό σφάλμα; Εάν δοκιμάζαμε αριθμητικές πεπερασμένες διαφορές, θα χρειαζόμασταν $N + 1$ εμπρόσθιες διελεύσεις σε κάθε βήμα, κάτι υπολογιστικά αδύνατο. Η **Οπισθοδιάδοση Σφάλματος (Reverse-Mode Autodiff / Backpropagation)** εφαρμόζει τον **Κανόνα της Αλυσίδας** από την έξοδο προς την είσοδο σε **μία μόνο εμπρόσθια και μία μόνο ανάστροφη διέλευση**, αποθηκεύοντας προσωρινά τις ενεργοποιήσεις (z, a) στη μνήμη και ανακυκλώνοντας την ανοδική κλίση $\partial L/\partial z$ για κάθε εισερχόμενο βάρος.

$$L_{BCE} = -(1/N)\,\Sigma\,[\,y_i \ln(\hat{y}_i) + (1 - y_i)\ln(1 - \hat{y}_i)\,] \quad | \quad \partial L/\partial w_i = (\partial L/\partial \hat{y}) \cdot (\partial \hat{y}/\partial z) \cdot (\partial z/\partial w_i) = (\hat{y} - y) \cdot x_i$$

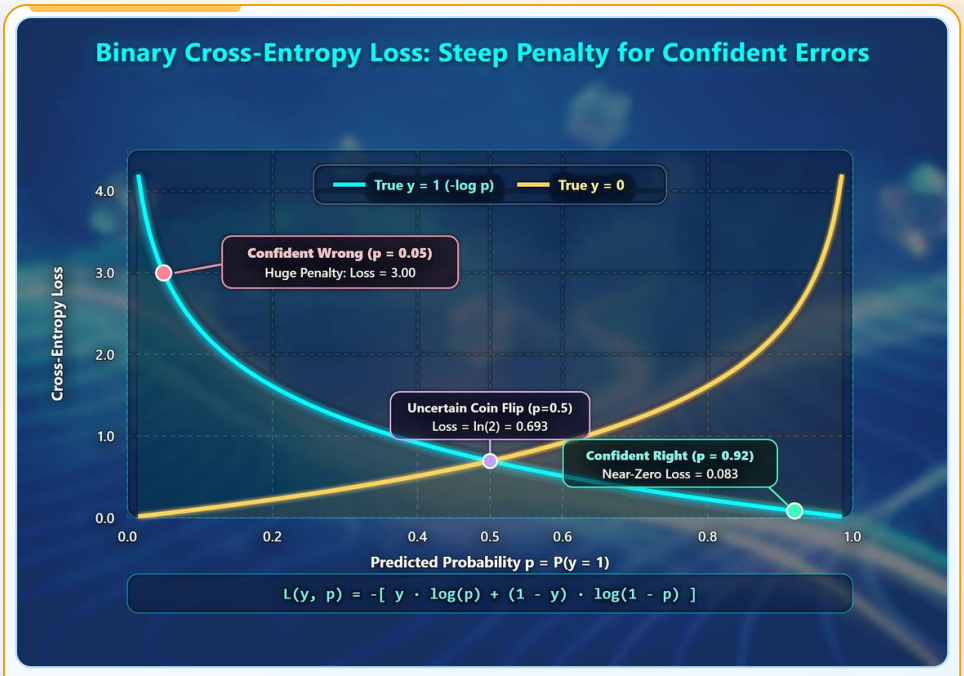
Αλγεβρική απλοποίηση BCE + Sigmoid: το γινόμενο $(\partial L/\partial \hat{y}) = -y/\hat{y} + (1-y)/(1-\hat{y})$ με τη $(\partial \hat{y}/\partial z) = \hat{y}(1-\hat{y})$ απαλείφει τους παρονομαστές, δίνοντας την καθαρή, αριθμητικά ευσταθή κλίση $\partial L/\partial z = \hat{y} - y$.

Ας παρακολουθήσουμε βήμα προς βήμα το αριθμητικό παράδειγμα του **Σχήματος 3.2** για στόχο $y = 1.0$ με εισόδους $x_1 = 1.0$, $x_2 = 0.5$, βάρη $w_1 = 0.80$, $w_2 = -0.40$ και πόλωση $b = 0.20$. Στην εμπρόσθια διέλευση προκύπτει γραμμικό άθροισμα $z = 0.80 - 0.20 + 0.20 = 0.80$, ενεργοποίηση $a = \sigma(0.80) = 0.690$ και απώλεια $L = -\ln(0.690) = 0.371$. Στην ανάστροφη διέλευση διαδίδουμε τις τοπικές παραγώγους: $\partial L/\partial a = -1/0.690 = -1.449$ και $\partial a/\partial z = 0.690 \cdot (1 - 0.690) = 0.214$, των οποίων το γινόμενο δίνει $\partial L/\partial z = -0.310 = a - y$. Πολλαπλασιάζοντας με κάθε είσοδο λαμβάνουμε $\partial L/\partial w_1 = -0.310$, $\partial L/\partial w_2 = -0.155$ και $\partial L/\partial b = -0.310$. Το αρνητικό πρόσημο της κλίσης σημαίνει ότι το βήμα καθόδου $w_{new} = w - \eta(\partial L/\partial w)$ αυξάνει τα w_1, w_2, b, ωθώντας τη σιγμοειδή πρόβλεψη a πλησιέστερα στον στόχο 1.0.



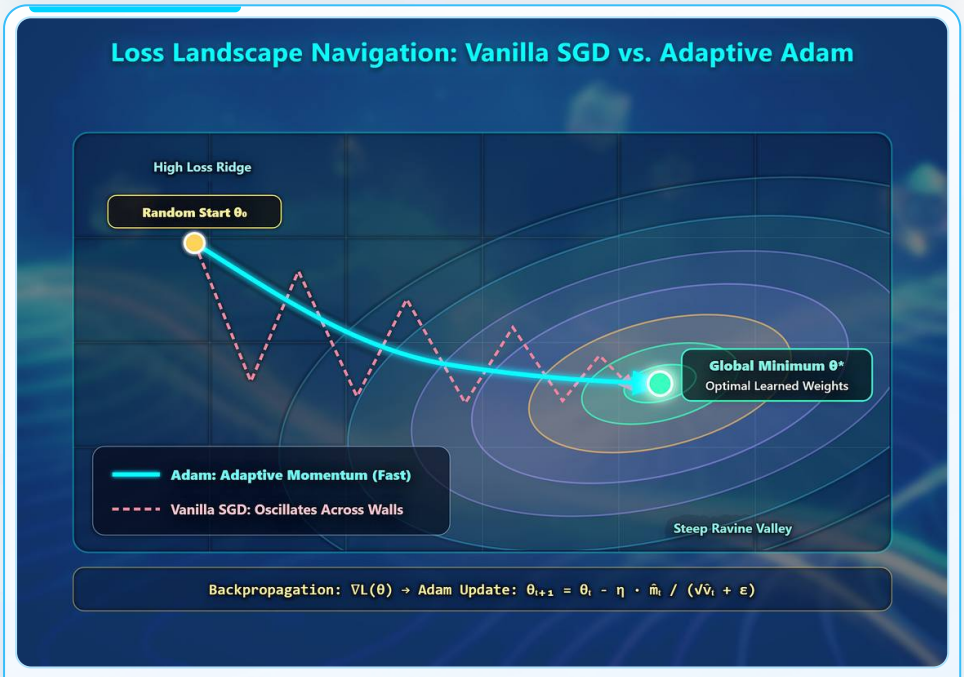
Σχήμα 3.2

Υπολογιστικός γράφος εμπρόσθιας διέλευσης και αντίστροφης αυτόματης διαφόρισης (Backpropagation) με αριθμητικό παράδειγμα και σύγκριση Adam+L2 έναντι AdamW.



Σχήμα 3.3α

Συμπεριφορά της συνάρτησης απώλειας Binary Cross-Entropy (BCE) για λανθασμένες και ορθές προβλέψεις.



Σχήμα 3.3β

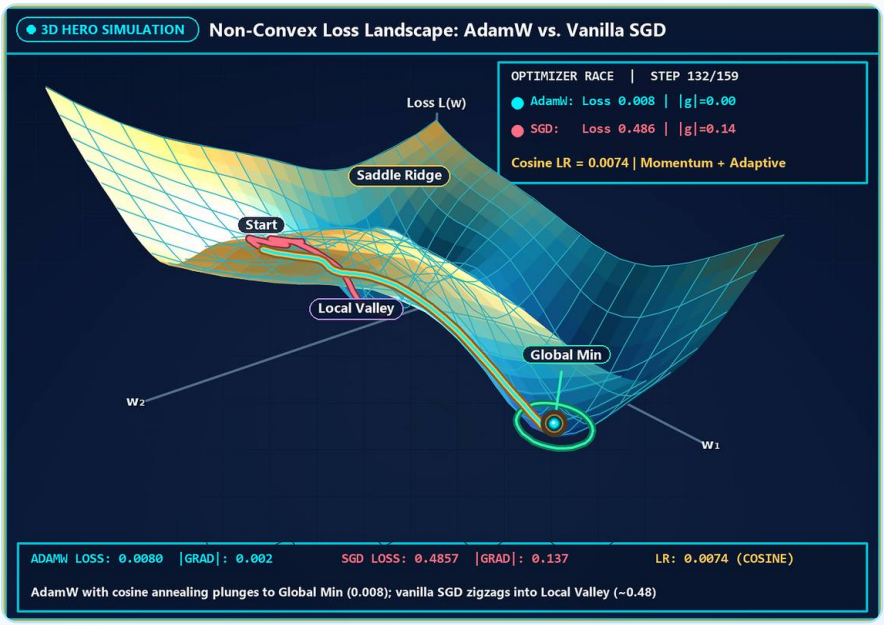
Τοπίο απώλειας, κάθοδος κλίσης (SGD) και προσαρμοστική ορμή του βελτιστοποιητή Adam.

Επειδή το τοπίο απώλειας των βαθιών νευρωνικών δικτύων είναι **μη κυρτό** (γεμάτο στενές κοιλάδες, πλατώ και σημεία σέλας), η απλή Στοχαστική Κάθοδος Κλίσης (SGD) συχνά ταλαντώνεται στα απότομα τοιχώματα ή επιβραδύνεται σε σέλες. Οι σύγχρονοι βελτιστοποιητές **Adam** και **AdamW** (*Loshchilov & Hutter, 2019*) συνδυάζουν *ορμή* (εκθετικά σταθμισμένο μέσο 1ης ροπής m_t) με *προσαρμοστικό ρυθμό μάθησης ανά παράμετρο* (2η ροπή v_t) και **διόρθωση μεροληψίας** στα αρχικά βήματα. Κρίσιμη υπεροχή του **AdamW** είναι η **αποσυνδεδεμένη φθορά βαρών (Decoupled Weight Decay)**: ενώ στον κλασικό Adam+L2 ο όρος $\lambda\theta_t$ ενσωματώνεται στην κλίση και διαιρείται ανομοιόμορφα από το $\sqrt{\hat{v}_t}$, ο AdamW εφαρμόζει τη συρρίκνωση $(1 - \eta\lambda)\theta_t$ απευθείας στα βάρη, προσφέροντας ανώτερη γενίκευση σε CNNs, Vision Transformers και LLMs.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad | \quad v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad | \quad \hat{m}_t = m_t/(1-\beta_1^t), \quad \hat{v}_t = v_t/(1-\beta_2^t)$$

$$\text{AdamW:} \quad \theta_{t+1} = (1 - \eta\lambda)\theta_t - \eta \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \varepsilon)$$

Τυπικές υπερπαράμετροι: $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$. Χωρίς διόρθωση μεροληψίας $(1-\beta^t)$, οι ροπές m_t, v_t θα ήταν τεχνητά μεροληπτικές προς το μηδέν στα πρώτα βήματα.

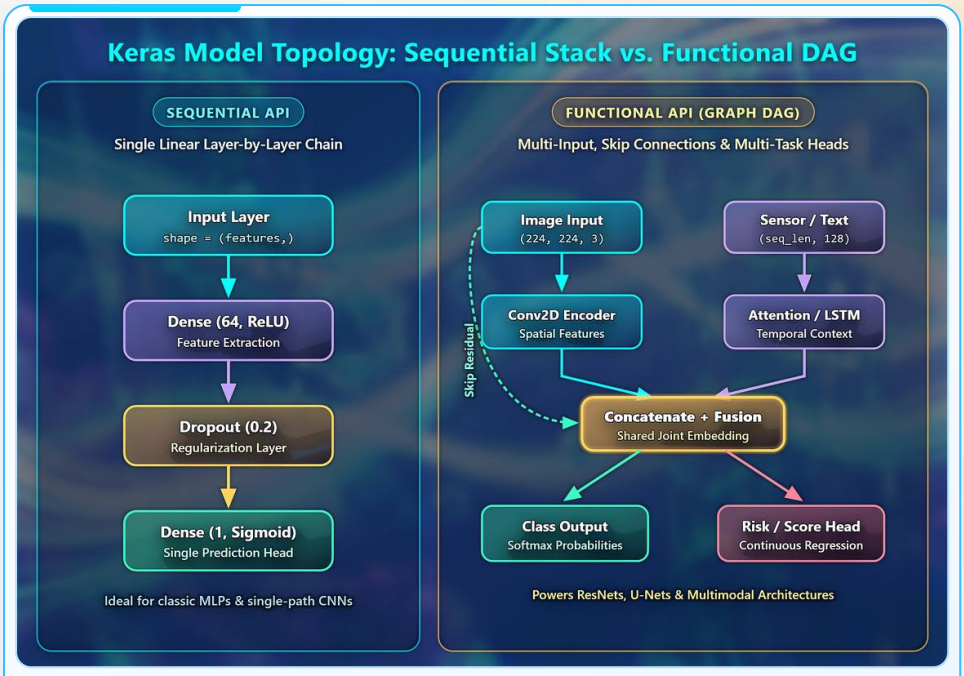


Σχήμα 3.4

Τρισδιάστατη προσομοίωση μη κυρτού τοπίου απώλειας: ο AdamW με συνημιτονοειδή απόσβεση ρυθμού μάθησης υπερβαίνει τη σέλα και φθάνει στο καθολικό ελάχιστο (0.008), ενώ ο απλός SGD παγιδεύεται σε τοπική κοιλάδα (0.486).

3.3 Keras 3: Sequential API έναντι Functional API & Παρακαμπτήριες Συνδέσεις

Στο **Keras 3** (το οποίο εκτελεί ενιαίο κώδικα πάνω σε TensorFlow, PyTorch ή JAX), η δόμηση μοντέλων υποστηρίζει δύο συμπληρωματικά προγραμματιστικά πρότυπα: το **Sequential API** για ευθύγραμμες στοίβες επιπέδων και το **Functional API** για γενικούς κατευθυνόμενους άκυκλους γράφους (*DAG*) με πολλαπλές εισόδους, διακλαδώσεις και **Παρακαμπτήριες Συνδέσεις (Residual Skip Connections)**. Μαθηματικά, ένα υπολειμματικό μπλοκ $y = x + F(x)$ διαθέτει παράγωγο $\partial y/\partial x = I + \partial F/\partial x$: ακόμη και όταν οι κλίσεις του εσωτερικού κλάδου $\partial F/\partial x$ γίνουν πολύ μικρές, ο μοναδιαίος όρος I εξασφαλίζει απρόσκοπτο «αυτοκινητόδρομο κλίσης» (*gradient highway*), καθιστώντας εφικτή την εκπαίδευση δικτύων βάθους δεκάδων ή εκατοντάδων επιπέδων (ResNet, Transformers).



Σχήμα 3.5

Αρχιτεκτονική σύγκριση Keras Sequential API (γραμμική ροή) και Functional API (κατευθυνόμενος γράφος με Residual Skip Connection).

ΕΡΓΑΣΤΗΡΙΟ 03 **Πύλη AND με Keras 3, Βελτιστοποιητή AdamW, EarlyStopping & Αποθήκευση .keras**

`code/03_and_gate_keras.py`

Μεταφέρουμε τη λογική πύλη AND στο σύγχρονο **Keras 3** με αρχιτεκτονική `Dense(4, relu) → Dense(1, sigmoid)`, μεταγλώττιση με **AdamW**, αυτόματη διακοπή `EarlyStopping(restore_best_weights=True)` και εγγενή σειριοποίηση σε αρχείο `gate_and.keras` με άμεση επαλήθευση επαναφόρτωσης.

Προτεινόμενος Πειραματισμός: Ανοίξτε το παραγόμενο αρχείο `gate_and.keras` στο εργαλείο **Netron** (Κεφάλαιο 7) για να επιθεωρήσετε οπτικά τον γράφο των τανυστών και τις διαστάσεις των πινάκων βαρών.

PYTHON 3.12+ `code/03_and_gate_keras.py` Πύλη AND με Keras 3, AdamW & εγγενή αποθήκευση .keras

```python
from pathlib import Path
import numpy as np
import keras
from keras import layers


def build_and_model() -> keras.Model:
    """Δημιουργεί και μεταγλωττίζει συμπαγές μοντέλο Sequential για τη λογική πύλη AND."""
    model = keras.Sequential(
        [
            keras.Input(shape=(2,), name="binary_inputs"),
            layers.Dense(4, activation="relu", name="hidden_dense"),
            layers.Dense(1, activation="sigmoid", name="probability_output"),
        ],
        name="gate_and_classifier",
    )

    # Μεταγλώττιση με AdamW (αποσυνδεδεμένη φθορά βαρών) και δυαδική διασταυρούμενη εντροπία
    model.compile(
        optimizer=keras.optimizers.AdamW(learning_rate=0.05),
        loss="binary_crossentropy",
        metrics=["accuracy"],
    )
    return model
```

```python
def main() -> None:
    """Εκπαιδεύει, αποθηκεύει και επαναφορτώνει το μοντέλο Keras 3 για την πύλη AND."""
    keras.utils.set_random_seed(42)

    # === 1. Ορισμός Δεδομένων Εκπαίδευσης (Πύλη AND) ===
    X_train = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
    y_train = np.array([[0], [0], [0], [1]], dtype=np.float32)

    # === 2. Κατασκευή & Εκπαίδευση Μοντέλου με EarlyStopping ===
    model = build_and_model()
    model.summary()

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/03_and_gate_keras.py του code.zip)
```

ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
Αποθήκευση gate_and.keras | Προβλέψεις: ['[0, 0]->0.002', '[0, 1]->0.029', '[1, 0]->0.029', '[1, 1]->0.967']
```

ΕΡΓΑΣΤΗΡΙΟ 05 Σύγκριση Sequential API & Functional API με Residual Skip Connection

`code/05_sequential_vs_functional_api.py`

Αντιπαραβάλλει άμεσα έναν γραμμικό ταξινομητή 3 κλάσεων (8 χαρακτηριστικά εισόδου, ενεργοποιήσεις **GELU** και έξοδο **Softmax**) με ένα μοντέλο **Functional API** που ενσωματώνει υπολειμματική παρακαμπτήρια άθροιση `layers.Add()([projected, branch])` και `LayerNormalization`, επιβεβαιώνοντας ότι κάθε έξοδος Softmax αθροίζει ακριβώς σε 1.0000.

Προτεινόμενος Πειραματισμός: Συγκρίνετε τα `model.summary()` των δύο αρχιτεκτονικών: παρατηρήστε στη στήλη *Connected to* πώς το Functional API καταγράφει ρητά τις δύο εισερχόμενες συνδέσεις στον κόμβο `skip_add`.

PYTHON 3.12+ `code/05_sequential_vs_functional_api.py` Δόμηση Sequential vs Functional Residual Graph στο Keras 3

```python
import numpy as np
import keras
from keras import layers


def build_sequential_model(input_features: int = 8, num_classes: int = 3) -> keras.Model:
    """Κατασκευάζει γραμμική στοίβα πυκνών επιπέδων (GELU -> Softmax) με το Sequential API."""
    return keras.Sequential(
        [
            keras.Input(shape=(input_features,), name="seq_input"),
            layers.Dense(32, activation="gelu", name="seq_dense_1"),
            layers.Dense(32, activation="gelu", name="seq_dense_2"),
            layers.Dense(num_classes, activation="softmax", name="seq_output"),
        ],
        name="sequential_classifier",
    )


def build_functional_residual_model(input_features: int = 8, num_classes: int = 3) -> keras.Model:
    """Κατασκευάζει μη γραμμικό γράφο με παρακαμπτήρια σύνδεση (Residual Connection) στο Functional
API."""
    inputs = keras.Input(shape=(input_features,), name="func_input")

    # Αρχική προβολή του διανύσματος εισόδου στον ενδιάμεσο χώρο 32 διαστάσεων
    projected = layers.Dense(32, activation="gelu", name="projection")(inputs)

    # Κύριος υπολειμματικός κλάδος F(x) δύο πυκνών επιπέδων
    branch = layers.Dense(32, activation="gelu", name="residual_dense_1")(projected)
    branch = layers.Dense(32, activation=None, name="residual_dense_2")(branch)

    # Παρακαμπτήρια άθροιση (Skip Connection: x + F(x)) για απρόσκοπτη ροή κλίσεων σε βαθιά δίκτυα
    merged = layers.Add(name="skip_add")([projected, branch])
    merged = layers.LayerNormalization(name="layer_norm")(merged)
    merged = layers.Activation("gelu", name="post_norm_gelu")(merged)
```

```
    # Τελική προβολή σε κατανομή πιθανοτήτων Softmax στις num_classes κατηγορίες
    outputs = layers.Dense(num_classes, activation="softmax", name="func_output")(merged)
    return keras.Model(inputs=inputs, outputs=outputs, name="functional_residual_classifier")


def main() -> None:
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/05_sequential_vs_functional_api.py του code.zip)
```

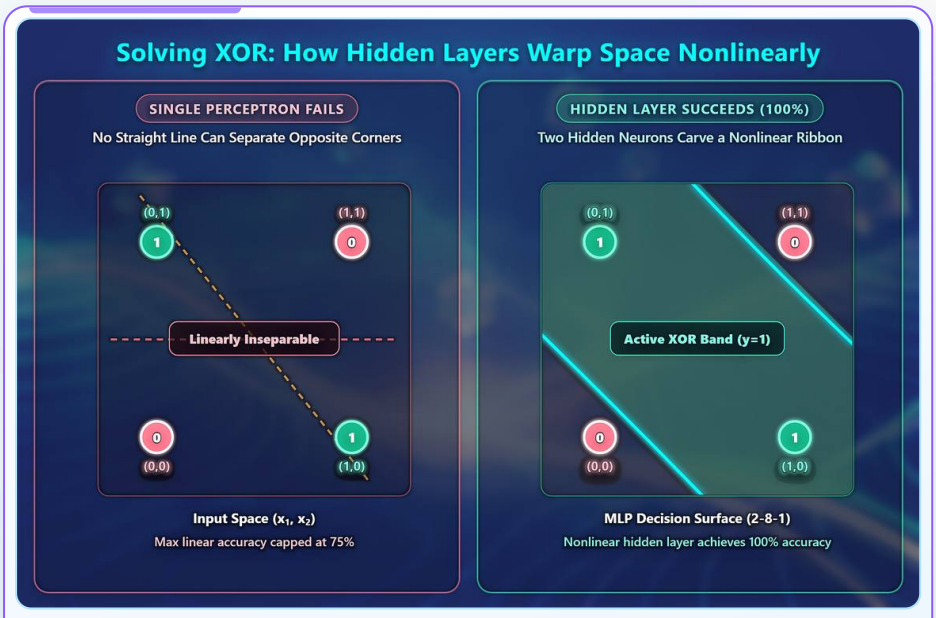
ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
Επαλήθευση αθροίσματος πιθανοτήτων Softmax (αναμενόμενο 1.0):
  Sequential API: [1. 1.]  |  Functional API: [1. 1.]
```

3.4 Το Ιστορικό Πρόβλημα XOR, Θεώρημα Καθολικής Προσέγγισης & Εργαστήρια 06 έως 08

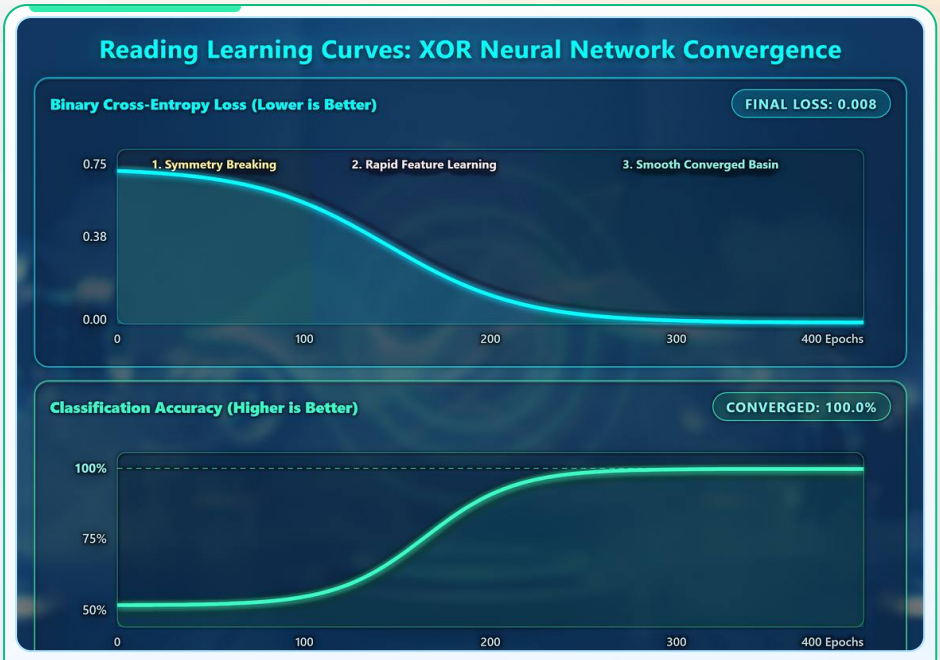
Το 1969 οι Minsky και Papert απέδειξαν αυστηρά ότι ένα απλό Perceptron ενός επιπέδου αδυνατεί να επιλύσει τη λογική πύλη **XOR** (αποκλειστικό Ή), όπου `(0,0), (1,1) → 0` ενώ `(0,1), (1,0) → 1`. Πράγματι, αν υπήρχε ευθεία $w_1x_1 + w_2x_2 + b = 0$ που τα διαχωρίζει, θα έπρεπε ταυτόχρονα $b < 0$ και $w_1 + w_2 + b < 0$ (άρα άθροισμα $w_1 + w_2 + 2b < 0$), ενώ για τα σημεία της κλάσης 1 απαιτείται $w_1 + b > 0$ και $w_2 + b > 0$ (άρα άθροισμα $w_1 + w_2 + 2b > 0$), καθαρή μαθηματική αντίφαση!

Η λύση αναδύεται από το **Θεώρημα Καθολικής Προσέγγισης (Universal Approximation Theorem, Cybenko 1989 & Hornik 1991)**: ένα νευρωνικό δίκτυο με έστω και *ένα* κρυφό επίπεδο επαρκούς πλάτους και μη πολυωνυμική ενεργοποίηση (`Tanh`, `ReLU`, `GELU`) μπορεί να προσεγγίσει οποιαδήποτε συνεχή συνάρτηση σε συμπαγές υποσύνολο του R^n με αυθαίρετη ακρίβεια $\varepsilon > 0$. Διαισθητικά, το κρυφό επίπεδο *αναδιπλώνει μη γραμμικά* τον δισδιάστατο χώρο εισόδου σε έναν νέο λανθάνοντα χώρο χαρακτηριστικών, μέσα στον οποίο τα τέσσερα σημεία της πύλης XOR γίνονται πλέον απολύτως γραμμικά διαχωρίσιμα από τον τελικό νευρώνα εξόδου!



Σχήμα 3.6α

Μη γραμμικό όριο απόφασης της πύλης XOR με πολυεπίπεδο νευρωνικό δίκτυο (MLP).



Σχήμα 3.6β

Καμπύλες σύγκλισης απώλειας (BCE) και ακρίβειας κατά την εκπαίδευση της πύλης XOR.

ΕΡΓΑΣΤΗΡΙΟ 06 **Επίλυση Μη Γραμμικού Προβλήματος XOR από το Μηδέν σε Καθαρή NumPy**

`code/06_xor_gate_numpy.py`

Υλοποιεί χειροκίνητα ένα MLP (`2 → 4 → 1`) με κρυφή ενεργοποίηση **Tanh** (παράγωγος $1 - a_1^2$), έξοδο **Sigmoid** και απώλεια **BCE**. Εκπαιδεύει επί 6.000 εποχές αναδεικνύοντας την ακριβή ροή των πινάκων κλίσης (`dz2, dW2, da1, dz1, dW1`) χωρίς κανένα πλαίσιο υψηλού επιπέδου.

Προτεινόμενος Πειραματισμός: Μειώστε το `hidden_dim` σε `2` (το θεωρητικό ελάχιστο για το XOR) και δοκιμάστε 3 διαφορετικά `seed` : παρατηρήστε γιατί 4 νευρώνες συγκλίνουν πολύ πιο αξιόπιστα αποφεύγοντας τοπικές παγίδες.

PYTHON 3.12+ `code/06_xor_gate_numpy.py` Πολυεπίπεδο δίκτυο MLP (2 -> 4 -> 1) σε καθαρή NumPy για XOR

```python
import numpy as np


def sigmoid(z: np.ndarray) -> np.ndarray:
    """Σιγμοειδής συνάρτηση ενεργοποίησης για παραγωγή πιθανότητας στο (0, 1)."""
    return 1.0 / (1.0 + np.exp(-np.clip(z, -500.0, 500.0)))


def binary_cross_entropy(y_true: np.ndarray, y_pred: np.ndarray) -> float:
    """Υπολογίζει τη δυαδική διασταυρούμενη εντροπία (BCE) με προστασία από log(0)."""
    eps = 1e-12
    p = np.clip(y_pred, eps, 1.0 - eps)
    return float(-np.mean(y_true * np.log(p) + (1.0 - y_true) * np.log(1.0 - p)))


def main() -> None:
    """Εκπαιδεύει από το μηδέν πολυεπίπεδο δίκτυο NumPy στη λογική πύλη XOR."""
    # === 1. Προετοιμασία Πίνακα Αλήθειας XOR ===
    rng = np.random.default_rng(seed=2026)

    X = np.array(
        [
            [0.0, 0.0],
            [0.0, 1.0],
            [1.0, 0.0],
            [1.0, 1.0],
        ],
```

```
        dtype=np.float64,
    )
    y = np.array([[0.0], [1.0], [1.0], [0.0]], dtype=np.float64)

    # === 2. Αρχικοποίηση Παραμέτρων MLP (2 -> 4 -> 1) ===
    # Κλιμάκωση αρχικών βαρών ανάλογα με 1 / sqrt(fan_in) για ομαλή είσοδο στην Tanh
    input_dim, hidden_dim, output_dim = 2, 4, 1
    W1 = rng.normal(0.0, np.sqrt(1.0 / input_dim), size=(input_dim, hidden_dim))
    b1 = np.zeros((1, hidden_dim), dtype=np.float64)
    W2 = rng.normal(0.0, np.sqrt(1.0 / hidden_dim), size=(hidden_dim, output_dim))
    b2 = np.zeros((1, output_dim), dtype=np.float64)

    learning_rate = 0.25
    epochs = 6_000
    n_samples = X.shape[0]
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/06_xor_gate_numpy.py του code.zip)
```

ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
Εποχή 1 | BCE: 0.6941  ->  Εποχή 6000 | BCE: 0.0017
(0,0)->0.0012 [0] | (0,1)->0.9981 [1] | (1,0)->0.9982 [1] | (1,1)->0.0021 [0]
```

ΕΡΓΑΣΤΗΡΙΟ 07 **Εκπαίδευση Πύλης XOR σε Keras 3 (GELU, EarlyStopping) & Οπτικοποίηση Σύγκλισης**

`code/07_xor_gate_keras.py`

Κατασκευάζει δίκτυο `Dense(8, gelu) → Dense(1, sigmoid)` με **Adam (lr = 0.05)** και **EarlyStopping** (*patience = 60*), σχεδιάζει τις καμπύλες απώλειας/ακρίβειας και αποθηκεύει το εκπαιδευμένο μοντέλο στο `xor_gate_model.keras`.

Προτεινόμενος Πειραματισμός: Αντικαταστήστε προσωρινά την ενεργοποίηση `gelu` με `linear` και εκτελέστε εκ νέου το πρόγραμμα: η ακρίβεια θα εγκλωβιστεί στο 50%, αποδεικνύοντας εμπειρικά την αδυναμία γραμμικού διαχωρισμού.

PYTHON 3.12+ `code/07_xor_gate_keras.py` — Εκπαίδευση πύλης XOR με Keras 3 (GELU, EarlyStopping)

```
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras import layers


def main() -> None:
    """Εκπαιδεύει δίκτυο Keras 3 για την πύλη XOR, το αποθηκεύει και σχεδιάζει καμπύλες σύγκλισης."""
    keras.utils.set_random_seed(42)

    # === 1. Προετοιμασία Δεδομένων XOR & Ορισμός Μοντέλου ===
    X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
    y = np.array([[0], [1], [1], [0]], dtype=np.float32)

    model = keras.Sequential(
        [
            keras.Input(shape=(2,), name="xor_input"),
            layers.Dense(8, activation="gelu", name="hidden_representation"),
            layers.Dense(1, activation="sigmoid", name="xor_output"),
        ],
        name="xor_solver_keras",
    )

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.05),
        loss="binary_crossentropy",
        metrics=["accuracy"],
    )

    # === 2. Εκπαίδευση με EarlyStopping ===
    # Διακοπή της εκπαίδευσης αν η απώλεια δεν βελτιωθεί για 60 συνεχόμενες εποχές
    early_stop = keras.callbacks.EarlyStopping(
        monitor="loss", patience=60, min_delta=1e-4, restore_best_weights=True
```

```
    )

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/07_xor_gate_keras.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 08

Φόρτωση Αποθηκευμένου Μοντέλου .keras & Παραγωγική Εξαγωγή Συμπερασμάτων (Inference)

`code/08_xor_load_and_predict.py`

Επιδεικνύει τη θεμελιώδη βιομηχανική αρχή του διαχωρισμού **Εκπαίδευσης (Training)** από την **Εξαγωγή Συμπερασμάτων (Inference)**: φορτώνει το σειριοποιημένο `xor_gate_model.keras` του Εργαστηρίου 07 και εκτελεί εμπρόσθια διέλευση με `model(sample, training=False)` (πολύ ταχύτερη από την `predict` σε μεμονωμένα δείγματα και με απενεργοποιημένα Dropout/BatchNorm).

Προτεινόμενος Πειραματισμός: Δοκιμάστε ενδιάμεσες συνεχείς εισόδους όπως `0.5 0.5` ή `0.9 0.1` για να χαρτογραφήσετε πειραματικά τη συνεχή επιφάνεια πιθανότητας που έμαθε το δίκτυο.

PYTHON 3.12+ **code/08_xor_load_and_predict.py** — Φόρτωση xor_gate_model.keras & διαδραστικός βρόχος inference

```python
from pathlib import Path
import numpy as np
import keras


def main() -> None:
    """Φορτώνει το αποθηκευμένο μοντέλο XOR και εκτελεί διαδραστική εξαγωγή συμπερασμάτων."""
    # === 1. Έλεγχος Ύπαρξης & Φόρτωση Αποθηκευμένου Μοντέλου ===
    model_path = Path("xor_gate_model.keras")
    if not model_path.exists():
        print(
            f"Δεν βρέθηκε το αρχείο {model_path.name}. "
            "Εκτελέστε πρώτα το πρόγραμμα 07_xor_gate_keras.py για να δημιουργηθεί."
        )
        return

    model = keras.models.load_model(model_path)
    print(f"Επιτυχής φόρτωση μοντέλου: {model_path.resolve()}")
    print("Εισαγάγετε δύο τιμές (0 ή 1) χωρισμένες με κενό, ή «q» για έξοδο.\n")

    # === 2. Διαδραστικός Βρόχος Προβλέψεων (Inference) ===
    while True:
        raw = input("Είσοδοι (x1 x2): ").strip()
        if raw.lower() in ("q", "quit", "exit"):
            print("Τερματισμός προγράμματος.")
            break

        parts = raw.split()
        if len(parts) != 2:
            print("Παρακαλούμε δώστε ακριβώς δύο αριθμούς, π.χ. 1 0")
            continue

        try:
            x1, x2 = float(parts[0]), float(parts[1])
        except ValueError:
            print("Μη έγκυρη αριθμητική μορφή.")
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/08_xor_load_and_predict.py του code.zip)
```



ΚΕΦΑΛΑΙΟ 4

Βαθιές Αρχιτεκτονικές: MLP, CNN, RNN, LSTM και ConvLSTM2D

Πώς εξειδικεύονται τα βαθιά νευρωνικά δίκτυα ανάλογα με τη γεωμετρία των δεδομένων: μαθηματικά BatchNorm/Dropout, 2D συνέλιξη, πεδία υποδοχής, πλήρεις πύλες LSTM και χωροχρονικά ConvLSTM2D.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Ανάλυση της υπεροχής του **βάθους έναντι του πλάτους** στα Deep MLPs και των μαθηματικών μηχανισμών σταθεροποίησης **BatchNormalization** και **Inverted Dropout**.
- Πλήρης εξοικείωση με την αριθμητική της **2D Συνέλιξης** ($O = floor((W - K + 2P)/S) + 1$), τον **διαμοιρασμό βαρών** έναντι της έκρηξης παραμέτρων Dense και την ιεραρχική επέκταση του **πεδίου υποδοχής**.
- Αναλυτική κατανόηση και των **τεσσάρων πυλών της κυψέλης LSTM** (f_t, i_t, $\tilde{C}_t$, C_t, o_t, h_t) και του πώς το *Constant Error Carousel* επιλύει την εξασθένηση κλίσης.
- Επέκταση στις χωροχρονικές πύλες **ConvLSTM2D** για πενταδιάστατους τανυστές *(B, T, H, W, C)* σε μετεωρολογία και βίντεο.

4.1 Βαθύ Πολυεπίπεδο Δίκτυο (Deep MLP): Βάθος, BatchNormalization & Dropout

Ένα **Πολυεπίπεδο Νευρωνικό Δίκτυο (Multi-Layer Perceptron, MLP)** συνδέει διαδοχικά πυκνά επίπεδα (*Dense layers*) μεταξύ εισόδου και εξόδου. Παρότι το Θεώρημα Καθολικής Προσέγγισης εξασφαλίζει ότι ένα ρηχό αλλά τεράστιου πλάτους δίκτυο μπορεί θεωρητικά να αναπαραστήσει κάθε συνεχή συνάρτηση, το **βάθος** προσφέρει *εκθετικό πλεονέκτημα αποδοτικότητας*: κάθε επίπεδο επαναχρησιμοποιεί τα ενδιάμεσα χαρακτηριστικά των προηγούμενων επιπέδων, κατασκευάζοντας συνθετικές αναπαραστάσεις με τάξεις μεγέθους λιγότερες παραμέτρους.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Εικόνες, Χρόνος & το Σημειωματάριο του LSTM: Πώς Σκέφτεται Κάθε Αρχιτεκτονική

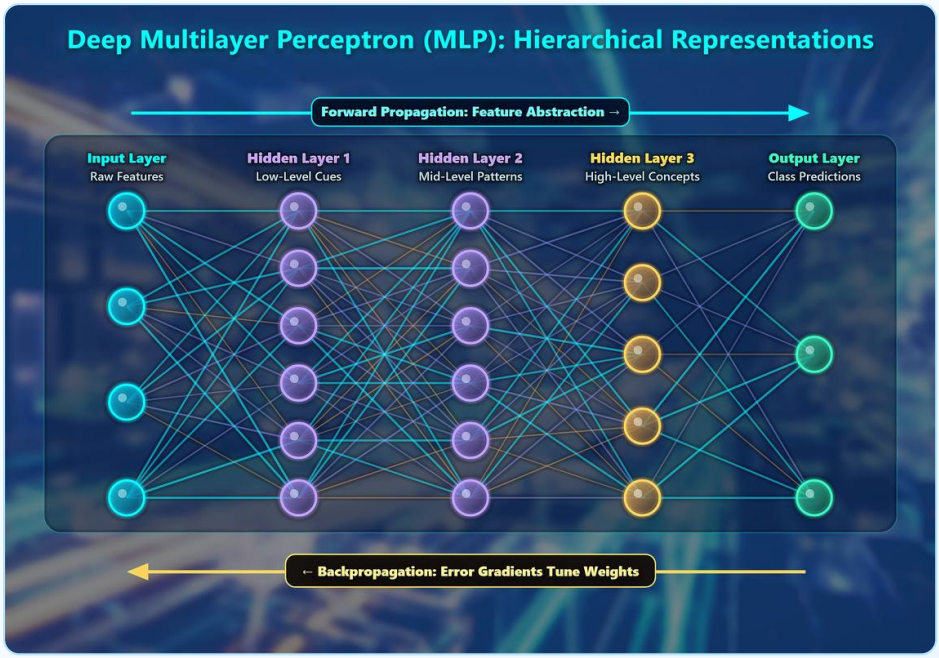
Δεν ταιριάζει το ίδιο εργαλείο σε κάθε μορφή δεδομένων. Κάθε οικογένεια βαθιών δικτύων εκμεταλλεύεται τη φυσική δομή του προβλήματος:

- **Deep MLP (Πίνακες):** Όλοι οι νευρώνες συνδέονται με όλους, ιδανικό για ανεξάρτητες αριθμητικές στήλες (ηλικία, πίεση, μετρήσεις).
- **CNN (Εικόνες & Χώρος):** Ένας μικρός «μεγεθυντικός φακός» (πυρήνας 3×3) σαρώνει τη φωτογραφία εντοπίζοντας πρώτα γραμμές και γωνίες, έπειτα υφές (μάτια, τροχούς) και τέλος ολόκληρα αντικείμενα οπουδήποτε στο κάδρο.
- **Το Σημειωματάριο του LSTM (Χρόνος & Κείμενο):** Φανταστείτε ότι διαβάζετε ένα βιβλίο κρατώντας ένα σημειωματάριο (*Κατάσταση Κυψέλης* C_t). Σε κάθε σελίδα χρησιμοποιείτε τρία εργαλεία: τη **γόμα** (**Πύλη Λήθης** f_t) για να σβήσετε παλιές ασήμαντες λεπτομέρειες, το **στυλό** (**Πύλη Εισόδου** i_t) για να γράψετε το νέο κρίσιμο γεγονός και τη **φωνή ανάγνωσης** (**Πύλη Εξόδου** o_t) για να απαντήσετε χρησιμοποιώντας μόνο ό,τι χρειάζεται τώρα!
- **Γλωσσάριο Τανυστών (Tensors):** Ένας τανυστής είναι γενικευμένος πίνακας: **1D** = διάνυσμα, **2D** = ασπρόμαυρη εικόνα, **3D** = έγχρωμη εικόνα `(H, W, 3)`, **4D** = δέσμη εικόνων `(B, H, W, C)`, **5D** = δέσμη βίντεο ή χαρτών καιρού `(B, T, H, W, C)`.

Για να εκπαιδεύονται σταθερά βαθιά δίκτυα χωρίς μετατόπιση της κατανομής των ενεργοποιήσεων και χωρίς απομνημόνευση θορύβου, χρησιμοποιούμε δύο θεμελιώδεις τεχνικές: την **Κανονικοποίηση Δέσμης (BatchNormalization)**, η οποία κεντράρει και κλιμακώνει τις ενεργοποιήσεις κάθε δέσμης B εξομαλύνοντας το τοπίο απώλειας, και το **Inverted Dropout**, το οποίο απενεργοποιεί τυχαία ένα ποσοστό p νευρώνων σε κάθε βήμα εκπαίδευσης, αναγκάζοντας το δίκτυο να λειτουργεί ως *σιωπηρό σύνολο (ensemble)* από 2^n ανθεκτικά υποδίκτυα.

$$\textbf{BatchNorm: } \hat{x}_i = (x_i - \mu_B) / \sqrt{(\sigma_B^2 + \varepsilon)}, \quad y_i = \gamma \hat{x}_i + \beta \quad | \quad \textbf{Dropout: } \tilde{h} = (m \odot h) / (1 - p), \quad m_j \sim \text{Bernoulli}(1 - p)$$

Οι εκπαιδεύσιμες παράμετροι γ, β επιτρέπουν στο BatchNorm να ανακτά το βέλτιστο εύρος τιμών, ενώ η διαίρεση με (1 − p) στο Inverted Dropout διατηρεί αμετάβλητη τη μέση τιμή ώστε στην εξαγωγή συμπερασμάτων να μη χρειάζεται καμία επανακλιμάκωση.



Σχήμα 4.1

Εσωτερική ανατομία βαθιού νευρωνικού δικτύου (είσοδος, κρυφά επίπεδα αναπαράστασης, συνδέσεις βαρών, έξοδος).

4.2 Συνελικτικά Νευρωνικά Δίκτυα (CNNs): Αριθμητική Συνέλιξης & Πεδία Υποδοχής

Φανταστείτε ότι επιχειρούμε να τροφοδοτήσουμε μια τυπική έγχρωμη εικόνα *224 × 224 × 3* (150.528 είσοδοι) σε ένα πλήρως συνδεδεμένο επίπεδο Dense που παράγει 64 χάρτες ίδιου μεγέθους: το πλήθος των εκπαιδεύσιμων βαρών θα εκτοξευόταν σε **δισεκατομμύρια παραμέτρους** (9.6 δισεκατομμύρια για έναν αντίστοιχο πλήρη χάρτη), προκαλώντας άμεση εξάντληση μνήμης και καταστροφική υπερεκπαίδευση, ενώ θα αγνοούσε εντελώς το γεγονός ότι γειτονικά εικονοστοιχεία συγκροτούν ακμές και υφές!

Τα **Συνελικτικά Νευρωνικά Δίκτυα (CNNs)** επιλύουν το πρόβλημα σαρώνοντας ολόκληρη την εικόνα με μικρά, *κοινόχρηστα* εκπαιδεύσιμα φίλτρα (**πυρήνες / kernels**, συνήθως *3 × 3*). Ένα επίπεδο **Conv2D** με 64 φίλτρα *3 × 3* πάνω στα 3 κανάλια RGB απαιτεί μόλις *(3 × 3 × 3) × 64 + 64 = 1.792 παραμέτρους* (μείωση άνω των **5.3 εκατομμυρίων φορών**!) και εξασφαλίζει **ισομεταβλητότητα σε μεταθέσεις**: ένα φίλτρο που έμαθε να εντοπίζει ένα μάτι ή μια γωνία στην άνω αριστερή περιοχή της εικόνας το αναγνωρίζει εξίσου άμεσα οπουδήποτε κι αν μετακινηθεί. Επιπλέον, η στοίβαξη δύο διαδοχικών επιπέδων *3 × 3* επιτυγχάνει ισοδύναμο **πεδίο υποδοχής (Receptive Field)** *5 × 5* με **28% λιγότερες παραμέτρους** (18 έναντι 25 ανά κανάλι) και μία επιπλέον μη γραμμικότητα ReLU! Για τους πρωτοετείς, οι τρεις ρυθμίσεις κάθε συνελικτικού επιπέδου έχουν άμεση φυσική ερμηνεία: το **Padding (*P*)** προσθέτει ένα λεπτό πλαίσιο μηδενικών γύρω από την εικόνα ώστε να μη συρρικνώνονται τα άκρα, το **Stride (*S*)** ορίζει το βήμα ολίσθησης του φίλτρου, ενώ το **Pooling** (π.χ. `MaxPooling 2×2`) κρατά τη μέγιστη τιμή κάθε τετράδας pixels, συμπιέζοντας τον χάρτη και καθιστώντας την αναγνώριση ανθεκτική σε μικρές μετατοπίσεις του αντικειμένου.

$$O = \lfloor (W - K + 2P) / S \rfloor + 1 \quad | \quad \text{Παράμετροι Conv2D} = (K_h \cdot K_w \cdot C_{in} + 1) \cdot C_{out} \quad | \quad RF_l = RF_{l-1} + (K_l - 1) \cdot \Pi_{i=1..l-1} S_i$$

Όπου W η χωρική διάσταση εισόδου, K το μέγεθος πυρήνα, P η συμπλήρωση περιθωρίου (padding: P=0 Valid, P=1 Same για K=3) και S το βήμα ολίσθησης (stride). Παράδειγμα: για W=5, K=3, P=0, S=1 προκύπτει O=3.

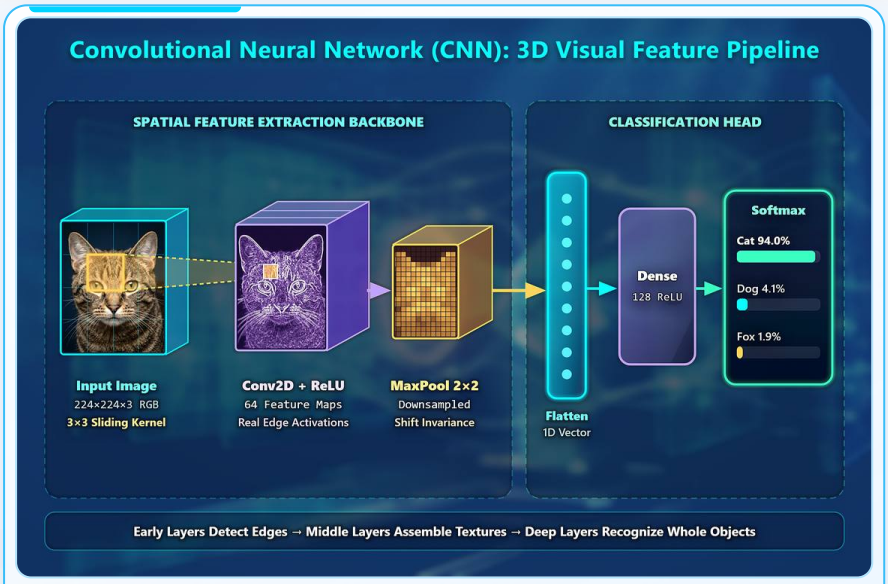
ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Εφαρμογή Πυρήνα Κάθετων Ακμών 3×3 σε Μικρο-Εικόνα 5×5 Βήμα προς Βήμα

Έστω μια ασπρόμαυρη εικόνα *5×5* με σκοτεινό αριστερό τμήμα (τιμή **0** στις στήλες 0, 1) και φωτεινό δεξιό τμήμα (τιμή **10** στις στήλες 2, 3, 4), δηλαδή κάθε γραμμή της είναι `[0, 0, 10, 10, 10]`. Εφαρμόζουμε τον κλασικό πυρήνα ανίχνευσης κάθετης ακμής *3×3* όπου κάθε γραμμή είναι `[-1, 0, +1]` (με $P = 0$, $S = 1$):

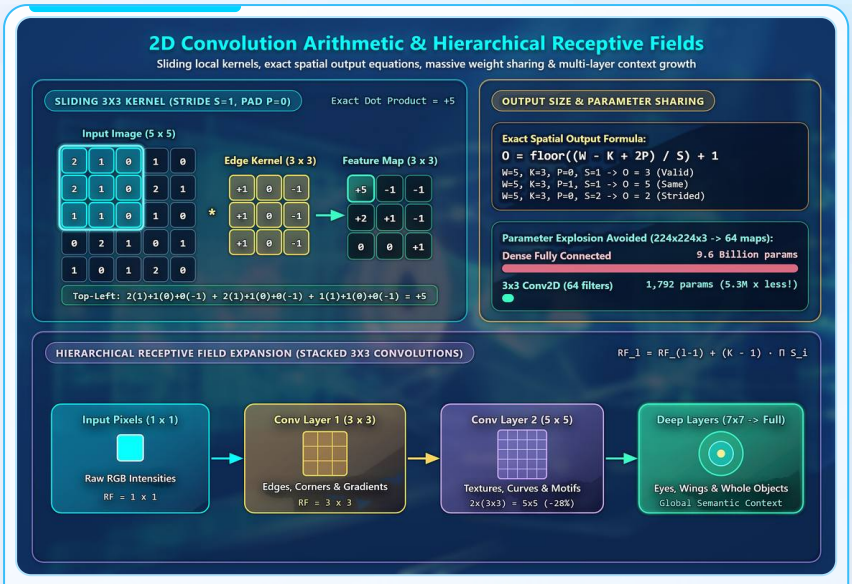
- **Βήμα 1 (Διάσταση εξόδου):** $O = \lfloor (5 - 3 + 0) / 1 \rfloor + 1 = \mathbf{3}$ → ο χάρτης χαρακτηριστικών εξόδου έχει σχήμα **3×3**.
- **Βήμα 2 (Θέση 0, στήλες 0 έως 2, παράθυρο `[0, 0, 10]`):** Σε κάθε μία από τις 3 γραμμές: $(-1)\times0 + 0\times0 + (+1)\times10 = +10$. Άθροισμα 3 γραμμών = **+30** (ισχυρή ακμή!).
- **Βήμα 3 (Θέση 1, στήλες 1 έως 3, παράθυρο `[0, 10, 10]`):** Σε κάθε γραμμή: $(-1)\times0 + 0\times10 + (+1)\times10 = +10$. Άθροισμα 3 γραμμών = **+30** (η ακμή παραμένει ισχυρή!).
- **Βήμα 4 (Θέση 2, στήλες 2 έως 4, επίπεδη φωτεινή περιοχή `[10, 10, 10]`):** Σε κάθε γραμμή: $(-1)\times10 + 0\times10 + (+1)\times10 = 0$. Άθροισμα = **0** (το ίδιο και σε επίπεδη σκοτεινή περιοχή `[0, 0, 0]`!).

Αποτέλεσμα 3×3: Κάθε γραμμή εξόδου είναι `[30, 30, 0]`! Οι ομοιόμορφες περιοχές σβήνουν στο 0, ενώ το σύνορο φωτός/σκιάς ανάβει στο +30. Στα CNNs το δίκτυο *μαθαίνει μόνο του* τα 9 βάρη κάθε πυρήνα μέσω Backpropagation.



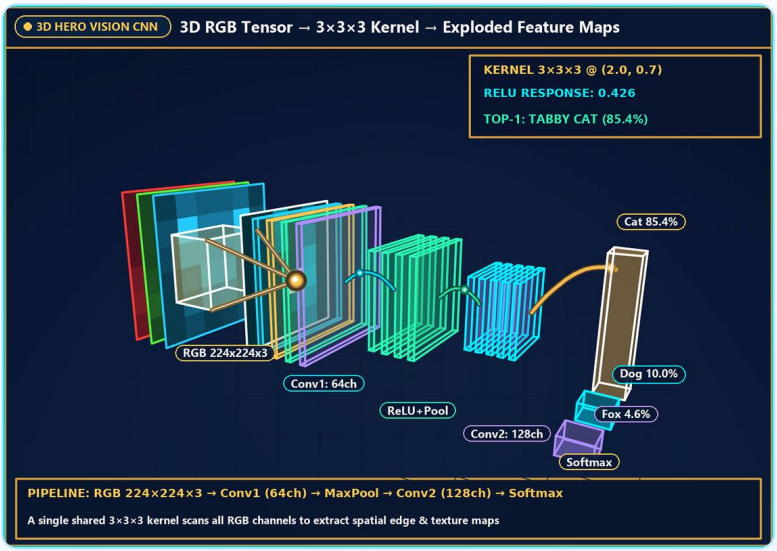
Σχήμα 4.2

Πλήρης ροή επεξεργασίας CNN από την αρχική εικόνα στους χάρτες χαρακτηριστικών (Conv2D + ReLU), το MaxPooling και την ταξινόμηση Softmax.



Σχήμα 4.3

Αριθμητική 2D συνέλιξης (πυρήνας 3x3), ακριβής εξίσωση διαστάσεων εξόδου, δραματική εξοικονόμηση παραμέτρων μέσω διαμοιρασμού βαρών και ιεραρχική επέκταση πεδίου υποδοχής.



Σχήμα 4.4

Τρισδιάστατη απεικόνιση σάρωσης RGB τανυστή (224x224x3) από κοινόχρηστο πυρήνα 3x3x3 προς ιεραρχικούς χάρτες χαρακτηριστικών (Conv1, ReLU+Pool, Conv2) και ταξινόμηση Softmax.

4.3 Αναδρομικά Δίκτυα (RNN), Πλήρεις Εξισώσεις LSTM & Χωροχρονικά ConvLSTM2D

Όταν τα δεδομένα οργανώνονται ως χρονικές ή γλωσσικές ακολουθίες (μετεωρολογικές χρονοσειρές, κείμενο, βιοσήματα), η διάταξη στον χρόνο φέρει κρίσιμη αιτιώδη πληροφορία. Ένα απλό **Αναδρομικό Νευρωνικό Δίκτυο (RNN)** ανανεώνει σε κάθε βήμα t την εσωτερική κρυφή κατάσταση $h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t + b_h)$. Κατά την **Οπισθοδιάδοση στον Χρόνο (BPTT)**, η κλίση από το βήμα T προς ένα παλαιότερο βήμα t περιέχει το επαναλαμβανόμενο γινόμενο $(W_{hh}^T)^{T-t}$: εάν η μέγιστη ιδιοτιμή του W_{hh} είναι μικρότερη του 1 οι κλίσεις σβήνουν εκθετικά (**Vanishing Gradients**, βραχύβια μνήμη), ενώ αν είναι μεγαλύτερη του 1 εκρήγνυνται (**Exploding Gradients**, αντιμετωπίζονται με αποκοπή κλίσης *Gradient Clipping*).

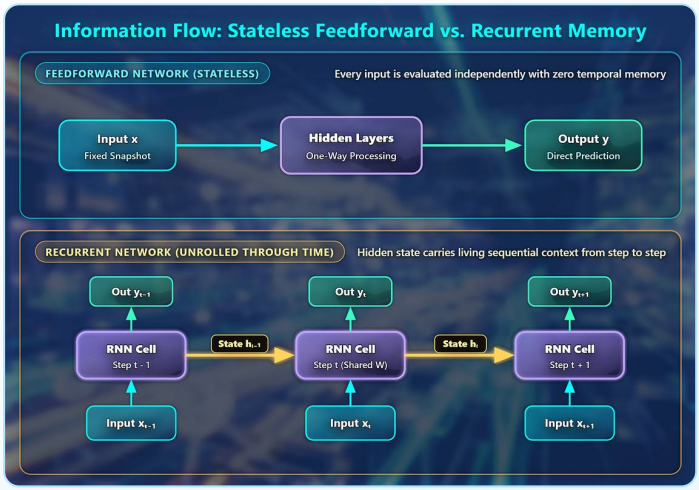
Η αρχιτεκτονική **LSTM (Long Short-Term Memory, Hochreiter & Schmidhuber, 1997)** επιλύει ριζικά τη βραχύβια μνήμη εισάγοντας έναν προστατευμένο δίαυλο μακρόχρονης μνήμης, την **Κατάσταση Κυψέλης (Cell State) C_t**, την οποία διαχειρίζονται τέσσερις διαφορίσιμες εξισώσεις πυλών. Διαισθητικά, η κυψέλη C_t λειτουργεί σαν ένας ιμάντας μεταφοράς πληροφορίας που διασχίζει ολόκληρη την ακολουθία: η **Πύλη Λήθης (f_t)** επιλέγει τι παρωχημένο διαγράφεται, η **Πύλη Εισόδου (i_t)** και η **Υποψήφια Κατάσταση ($\tilde{C}_t$)** προσθέτουν τη νέα χρήσιμη πληροφορία, ενώ η **Πύλη Εξόδου (o_t)** φιλτράρει τι εξάγεται στην τρέχουσα κρυφή κατάσταση h_t.

$$\mathbf{f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \;\; | \;\; i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \;\; | \;\; \tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)}$$

$$\mathbf{C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \;\; | \;\; o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \;\; | \;\; h_t = o_t \odot \tanh(C_t)}$$

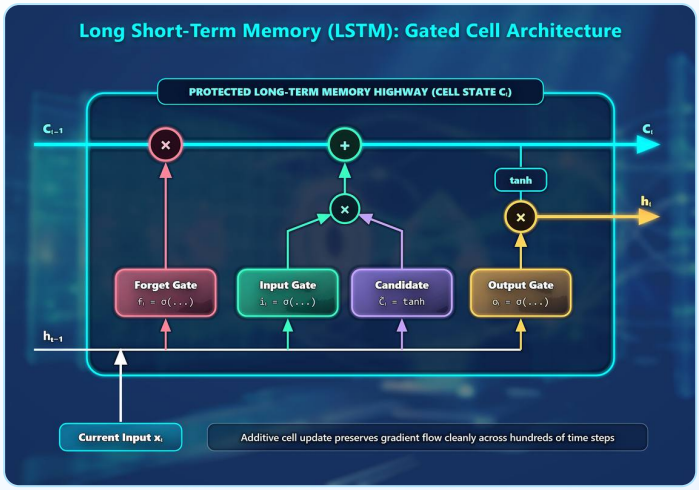
Μηχανισμός Constant Error Carousel: επειδή η C_t ενημερώνεται αθροιστικά (και όχι μέσω πολλαπλασιασμού με W_{hh}), η παράγωγος $\partial C_t / \partial C_{t-1} \approx f_t$ διατηρείται κοντά στη μονάδα όταν η πύλη λήθης παραμένει ανοιχτή ($f_t \approx 1$), επιτρέποντας την απρόσκοπτη διάδοση κλίσεων σε βάθος εκατοντάδων βημάτων!

Όταν το φυσικό σύστημα εξελίσσεται ταυτόχρονα στον **δισδιάστατο χώρο και στον χρόνο** (δορυφορικά νέφη, ραντάρ βροχόπτωσης, συζευγμένα μετεωρολογικά πλέγματα, καρέ βίντεο), η μετατροπή κάθε πλέγματος σε 1D διάνυσμα καταστρέφει τη χωρική τοπολογία. Το **ConvLSTM2D** αντικαθιστά όλους τους πολλαπλασιασμούς πινάκων εντός των τεσσάρων πυλών LSTM με δισδιάστατες συνελίξεις (`*`): $f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + b_f)$ (και αντίστοιχα για i_t, $\tilde{C}_t$, o_t), επεξεργαζόμενο απευθείας πενταδιάστατους τανυστές `(Batch, Time, Height, Width, Channels)`, αρχιτεκτονική που υλοποιούμε πρακτικά στην πρόγνωση καιρού Πάτρας στο **Εργαστήριο 17**.



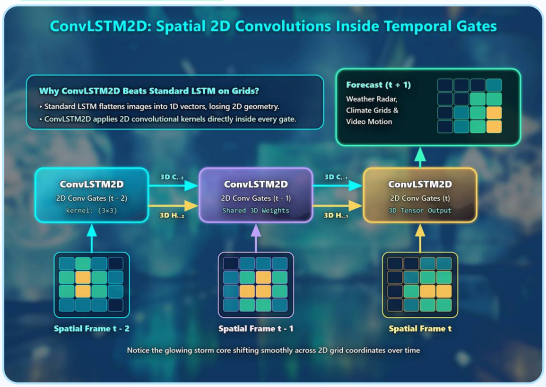
Σχήμα 4.5α

Feedforward δίκτυα έναντι Αναδρομικών Δικτύων (RNN) με χρονική αναδίπλωση.



Σχήμα 4.5β

Εσωτερική δομή κυψέλης LSTM (Πύλες Λήθης, Εισόδου, Εξόδου και αθροιστικός δίαυλος Κατάστασης Κυψέλης Ct).



Σχήμα 4.6

Αρχιτεκτονική ConvLSTM2D για χωροχρονική πρόβλεψη (π.χ. δυναμική εξέλιξη μετεωρολογικών πεδίων).

Αρχιτεκτονική	Διάσταση Εισόδου	Επαγωγική Προδιάθεση & Μηχανισμός	Κορυφαίο Πεδίο Εφαρμογής
Deep MLP	1D διάνυσμα χαρακτηριστικών (B × F)	Πλήρεις συνδέσεις Dense + BatchNorm/Dropout + ReLU/GELU	Πινακοποιημένα δεδομένα, λογικές πύλες, κεφαλές Q-Values (DQN)
CNN (2D)	4D τανυστής (B × H × W × C)	Χωρική τοπικότητα, διαμοιρασμός βαρών Conv2D + Pooling	Αναγνώριση εικόνων (MNIST, CIFAR-10), ανίχνευση αντικειμένων, φασματογράμματα
RNN / LSTM	3D τανυστής (B × T × F)	Χρονική αιτιότητα & Constant Error Carousel (4 πύλες μνήμης)	Χρονοσειρές, μετεωρολογία Πάτρας, ανάλυση συναισθήματος (Bi-LSTM)
ConvLSTM2D	5D τανυστής (B × T × H × W × C)	Συνελικτικές 2D πύλες μέσα σε αναδρομικό κύκλο LSTM	Χωροχρονική πρόγνωση καιρού, ραντάρ βροχόπτωσης, πρόβλεψη βίντεο

◆ **Πρακτικός Κανόνας Επιλογής Αρχιτεκτονικής (Inductive Bias)**

Η επιλογή αρχιτεκτονικής ξεκινά πάντα από τη **γεωμετρία των δεδομένων**: για ανεξάρτητα διανύσματα χαρακτηριστικών επιλέγουμε **MLP**, για εικόνες και χωρικά σήματα **CNN**, για ακολουθίες και χρονοσειρές **LSTM / Bi-LSTM** (ή Transformers για μεγάλα συμφραζόμενα), και για εξελισσόμενους χάρτες στον χρόνο **ConvLSTM2D**. Η σωστή επαγωγική προδιάθεση μειώνει δραματικά τα απαιτούμενα δεδομένα και μεγιστοποιεί την ικανότητα γενίκευσης.

ΚΕΦΑΛΑΙΟ 5

Αρχιτεκτονική Transformers και Μετωπικά Μοντέλα AI

Πλήρης μαθηματική παραγωγή Scaled Dot-Product & Multi-Head Self-Attention, Pre-LN Residual Connections, Κωδικοποιήσεις Θέσης (Sinusoidal, Learned, RoPE) και οικογένειες Encoder, Decoder και Sparse MoE.

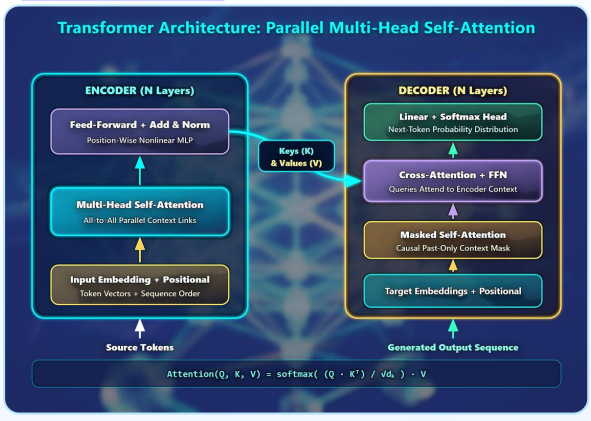
Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Κατανόηση του γιατί η **Αυτοπροσοχή (Self-Attention)** υπερβαίνει τον σειριακό περιορισμό των RNN/LSTM συνδέοντας οποιαδήποτε δύο tokens σε απόσταση $O(1)$.
- Αναλυτική παραγωγή της **Κλιμακωτής Αυτοπροσοχής (Scaled Dot-Product)** και του **Multi-Head Attention**, με στατιστική απόδειξη του ρόλου της διαίρεσης με $\sqrt{d_k}$.
- Διάκριση των **Pre-LN Residual Connections**, των σχημάτων **Κωδικοποίησης Θέσης** (*Sinusoidal, Learned, RoPE*) και των οικογενειών **Encoder-Only, Decoder-Only & Sparse MoE**.
- Εκτέλεση και επαλήθευση του πίνακα βαρών προσοχής σε καθαρή NumPy στο **Εργαστήριο 22** (`22_transformer_self_attention.py`).

5.1 Από τη Σειριακή Αναδρομή στη Μαθηματική Αυτοπροσοχή (Scaled Dot-Product & Multi-Head)

Μέχρι την ιστορική δημοσίευση «*Attention Is All You Need*» (Vaswani et al., 2017), η επεξεργασία ακολουθιών βασιζόταν σε αναδρομικά δίκτυα (RNN και LSTM) που διάβαζαν τις λέξεις αυστηρά σειριακά, μία προς μία, εμποδίζοντας την πλήρη αξιοποίηση των παράλληλων GPU και εξασθενώντας τις εξαρτήσεις μεγάλων αποστάσεων. Η αρχιτεκτονική **Transformer** κατάργησε πλήρως την αναδρομή και εισήγαγε τη **Κλιμακωτή Αυτοπροσοχή (Scaled Dot-Product Self-Attention)**.

Διαισθητικά, η αυτοπροσοχή λειτουργεί σαν μια *δυναμική, διαφορίσιμη μηχανή αναζήτησης μέσα στην ίδια την πρόταση*: κάθε λεκτική μονάδα (*token*) εκπέμπει ένα διάνυσμα **Ερωτήματος (Query, Q)** («ποια πληροφορία αναζητώ;»), διαφημίζει ένα **Κλειδί (Key, K)** («τι πληροφορία περιέχω;») και φέρει μια **Τιμή (Value, V)** («ποιο είναι το περιεχόμενό μου;»). Συγκρίνοντας ταυτόχρονα κάθε Q με όλα τα K μέσω ενός μοναδικού πολλαπλασιασμού πινάκων $Q \cdot K^T$, κάθε token συσχετίζεται άμεσα και παράλληλα με ολόκληρη την ακολουθία σε σταθερό βάθος διαδρομής $O(1)$.



Σχήμα 5.1

Αρχιτεκτονική Transformer: Κωδικοποίηση Θέσης, Multi-Head Self-Attention, Παρακαμπτήριες Συνδέσεις (Add & Norm) και Feed-Forward δίκτυο.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Αυτοπροσοχή (Self-Attention): Η Αναλογία της Βιβλιοθήκης & των Πολλαπλών Φακών

Διαβάστε την πρόταση: *«Ο σκύλος δεν πέρασε τον δρόμο γιατί **αυτός** ήταν πολύ κουρασμένος»*. Ο εγκέφαλός σας συνδέει ακαριαία την αντωνυμία *«αυτός»* με τη λέξη *«σκύλος»* (και όχι με τον *«δρόμο»*). Η **Αυτοπροσοχή** υλοποιεί ακριβώς αυτήν τη σύνδεση σαν μια έξυπνη βιβλιοθήκη: κάθε λέξη εκπέμπει ένα **Ερώτημα (Query, Q)** («τι αναζητώ;»), προβάλλει ένα **Κλειδί (Key, K)** («η ετικέτα του βιβλίου μου») και προσφέρει μια **Τιμή (Value, V)** («το περιεχόμενό μου»). Όσο καλύτερα ταιριάζει το *Q* με ένα *K*, τόσο μεγαλύτερο μερίδιο από το *V* του απορροφά η λέξη! Με το **Multi-Head Attention** χρησιμοποιούμε ταυτόχρονα πολλούς ανεξάρτητους «φακούς»: η μία κεφαλή εντοπίζει υποκείμενο-ρήμα, η δεύτερη αντωνυμίες και η τρίτη σημασιολογία.

- **Token:** Η βασική λεκτική μονάδα (λέξη ή υπολέξη/συλλαβή) στην οποία τεμαχίζεται το κείμενο.
- **Ενσωμάτωση (Embedding):** Αριθμητικό διάνυσμα που εκφράζει το νόημα ενός token σε πολυδιάστατο χώρο.
- **Πιθανότητες Softmax:** Μετατρέπουν τα σκορ ομοιότητας σε ποσοστά προσοχής που αθροίζουν ακριβώς στο 100% (1.0).
- **Κωδικοποίηση Θέσης (Positional Encoding):** Προσθέτει τη σειρά των λέξεων ώστε το μοντέλο να ξεχωρίζει το «Ο σκύλος κυνηγά τη γάτα» από το αντίστροφο.
- **Αιτιώδης Μάσκα (Causal Mask):** Κρύβει τις μελλοντικές λέξεις στα μοντέλα παραγωγής (GPT, Gemma) ώστε να προβλέπουν μόνο προς τα εμπρός.

Γιατί διαιρούμε υποχρεωτικά με $\sqrt{d_k}$; Ας υποθέσουμε ότι οι συνιστώσες των $q, k \in R^{d_k}$ είναι ανεξάρτητες τυχαίες μεταβλητές με μέση τιμή 0 και διακύμανση 1. Τότε το εσωτερικό τους γινόμενο $q \cdot k = \Sigma_{j=1..d_k} q_j k_j$ έχει μέση τιμή 0 αλλά **διακύμανση ίση με d_k** (τυπική απόκλιση $\sqrt{d_k}$). Για τυπική διάσταση κεφαλής $d_k = 128$, τα ακανονικοποίητα γινόμενα λαμβάνουν πολύ μεγάλες τιμές, ωθώντας τη συνάρτηση **Softmax** σε ακραίες περιοχές κορεσμού (πιθανότητες 0.9999 και 0.0000) όπου οι κλίσεις μηδενίζονται! Η διαίρεση με $\sqrt{d_k}$ επαναφέρει τη διακύμανση ακριβώς στο 1, διατηρώντας τις κλίσεις της Softmax υγιείς και σταθερές.

$$\mathbf{Attention(Q, K, V) = softmax((Q \cdot K^T) / \sqrt{d_k} + M) \cdot V}$$

$$\mathbf{MultiHead(Q, K, V) = Concat(head_1, ..., head_h) \cdot W^O, \quad head_i = Attention(X\,W_i^Q, X\,W_i^K, X\,W_i^V)}$$

Όπου $X \in R^{N \times d_{model}}$ ο πίνακας ενσωματώσεων, $d_k = d_{model}/h$ η διάσταση κάθε κεφαλής (π.χ. d_{model}=4096, h=32 ⇒ d_k=128) και M η αιτιώδης μάσκα ($-\infty$ άνω της διαγωνίου στα αυτοπαλίνδρομα μοντέλα).

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Αυτοπροσοχή στο Χέρι για την Πρόταση 3 Λέξεων [«Ο», «σκύλος», «τρέχει»]

Έστω η πρόταση 3 tokens `[«Ο», «σκύλος», «τρέχει»]` με διάσταση κεφαλής $d_k = 4$ (άρα παράγοντας κλιμάκωσης $\sqrt{d_k} = \sqrt{4} = 2.0$). Εξετάζουμε πώς το ρήμα **«τρέχει»** συγκεντρώνει συμφραζόμενα από τις τρεις λέξεις, όταν τα ακατέργαστα εσωτερικά γινόμενα $q_3 \cdot k_j$ είναι `[1.0, 2.0, 0.5]`:

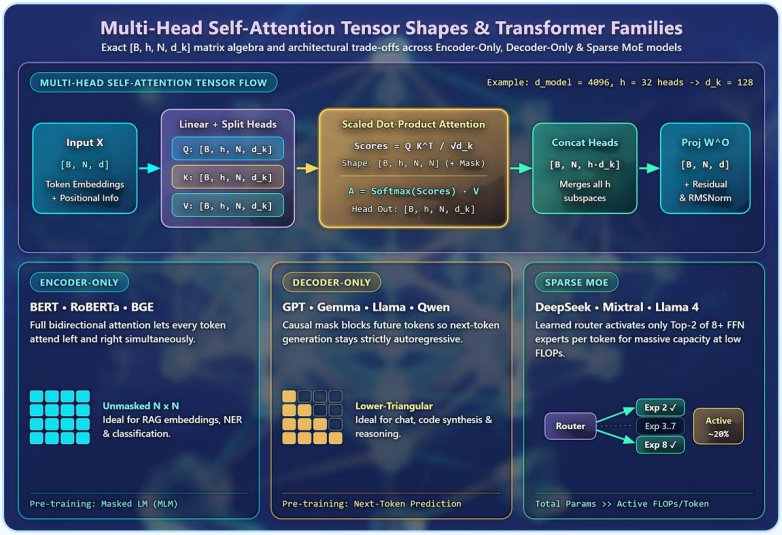
- **Βήμα 1 (Κλιμάκωση δια $\sqrt{d_k} = 2.0$):** `[1.0/2, 2.0/2, 0.5/2] = [0.50, 1.00, 0.25]`.
- **Βήμα 2 (Εκθετικά e^x & Πιθανότητες Softmax):** $e^{0.50} = 1.6487$, $e^{1.00} = 2.7183$, $e^{0.25} = 1.2840$ (άθροισμα *5.6510*). Διαιρώντας με το άθροισμα προκύπτουν τα βάρη προσοχής: **«Ο» = 0.292** (29,2%), **«σκύλος» = 0.481** (48,1%), **«τρέχει» = 0.227** (22,7%), με ακριβές άθροισμα *0.292 + 0.481 + 0.227 = 1.000*!
- **Βήμα 3 (Σταθμισμένος μέσος των διανυσμάτων Τιμής *V*):** Έστω $v_1 = [0.0, 1.0]$, $v_2 = [1.0, 0.0]$, $v_3 = [0.5, 0.5]$. Το νέο διάνυσμα εξόδου του «τρέχει» είναι: $0.292[0.0, 1.0] + 0.481[1.0, 0.0] + 0.227[0.5, 0.5] = \mathbf{[0.595, 0.405]}$.

Συμπέρασμα: Το ρήμα *«τρέχει»* άντλησε το 48,1% της νέας του αναπαράστασης από το υποκείμενό του (*«σκύλος»*) σε ένα μόνο παράλληλο βήμα! Έτσι ρέει το νόημα μέσα σε κάθε σύγχρονο LLM.

5.2 Pre-LN Συνδέσεις, Κωδικοποιήσεις Θέσης & Αρχιτεκτονικές Οικογένειες Transformer

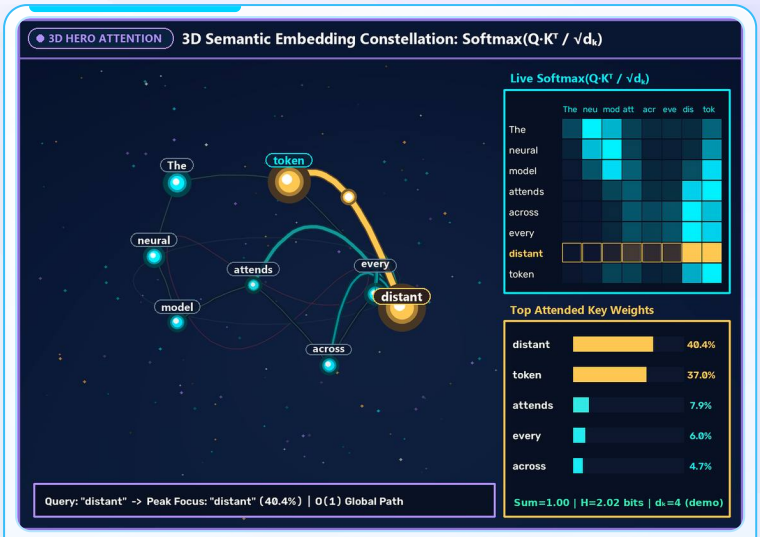
Πέρα από την ίδια την αυτοπροσοχή, τρεις μηχανικές επιλογές καθορίζουν τη σταθερότητα και την κλιμάκωση των σύγχρονων Transformers σε δεκάδες ή εκατοντάδες δισεκατομμύρια παραμέτρους:

- **Pre-LN Παρακαμπτήριες Συνδέσεις (Pre-LayerNorm / RMSNorm):** Ενώ το αρχικό άρθρο του 2017 εφάρμοζε κανονικοποίηση *μετά* την άθροιση (*Post-LN: LN(x + Sublayer(x))*), τα σύγχρονα LLMs χρησιμοποιούν **Pre-LN** (*x + Sublayer(RMSNorm(x))*). Με αυτόν τον τρόπο ο κύριος υπολειμματικός δίαυλος παραμένει εντελώς καθαρός από μετασχηματισμούς κανονικοποίησης, εξασφαλίζοντας σταθερές κλίσεις από την πρώτη κιόλας εποχή χωρίς εύθραυστη φάση προθέρμανσης.
- **Κωδικοποιήσεις Θέσης (Sinusoidal, Learned & RoPE):** Επειδή το $Q \cdot K^T$ είναι εγγενώς συμμετρικό ως προς τις μεταθέσεις (*permutation equivariant*), πρέπει να εγχύσουμε τη σειρά των λέξεων. Οι *Ημιτονοειδείς (Sinusoidal)* κωδικοποιήσεις προσθέτουν σταθερά κύματα διαφορετικών συχνοτήτων, οι *Εκπαιδεύσιμες (Learned)* μαθαίνουν ένα διάνυσμα ανά απόλυτη θέση (έως σταθερό μέγιστο μήκος), ενώ οι **Περιστροφικές (Rotary Positional Embeddings, RoPE)** περιστρέφουν τα διανύσματα *Q* και *K* στο μιγαδικό επίπεδο κατά γωνία ανάλογη της θέσης *m*, ώστε το εσωτερικό γινόμενο $q_m \cdot k_n$ να εξαρτάται αυστηρά από τη **σχετική απόσταση (m − n)**, επιτρέποντας παράθυρα εκατομμυρίων tokens.
- **Τρεις Αρχιτεκτονικές Οικογένειες:** Τα **Encoder-Only** μοντέλα (*BERT, RoBERTa, BGE*) χρησιμοποιούν πλήρη αμφίδρομη προσοχή χωρίς μάσκα (εκπαίδευση *Masked LM*) και υπερέχουν σε διανύσματα αναζήτησης RAG και ταξινόμηση. Τα **Decoder-Only** μοντέλα (*GPT, Gemma, Llama, Qwen*) επιβάλλουν κάτω τριγωνική αιτιώδη μάσκα για αυτοπαλίνδρομη πρόβλεψη επόμενου token (συνομιλία, κώδικας, συλλογισμός). Τέλος, τα **Sparse Mixture of Experts (MoE)** (*DeepSeek, Mixtral, Llama 4*) ενεργοποιούν μέσω εκπαιδεύσιμου δρομολογητή μόνο λίγους ειδικούς (π.χ. Top-2 από 8+ FFN, ~20% ενεργές παράμετροι ανά token), επιτυγχάνοντας τεράστια χωρητικότητα γνώσης με ελάχιστα FLOPs εξαγωγής συμπερασμάτων.



Σχήμα 5.2

Ροή διαστάσεων τανυστών [B, h, N, d_k] στο Multi-Head Self-Attention και σύγκριση των τριών αρχιτεκτονικών οικογενειών Transformer: Encoder-Only, Decoder-Only και Sparse MoE.



Σχήμα 5.3

Τρισδιάστατος αστερισμός σημασιολογικών ενσωματώσεων και ζωντανός πίνακας βαρών Softmax(Q K^T / sqrt(d_k)): άμεση διαδρομή O(1) μεταξύ απομακρυσμένων tokens.

5.3 Μετωπικά Μοντέλα AI: Συλλογισμός, MoE, Πολυτροπικότητα & Πράκτορες

Η κλιμάκωση των Transformers οδήγησε στη σύγχρονη γενιά **Θεμελιωδών Μοντέλων (Foundation Models)**, τα οποία χαρακτηρίζονται από τέσσερις κρίσιμες αρχιτεκτονικές καινοτομίες:

Μοντέλα Συλλογισμού (Test-Time Compute)

Μέσω ενισχυτικής μάθησης πάνω σε επαληθεύσιμα προβλήματα μαθηματικών, επιστήμης και κώδικα, τα μοντέλα αφιερώνουν επιπλέον υπολογιστικό χρόνο κατά την απάντηση (αλυσίδες σκέψης και αυτοδιόρθωση) για να επιλύουν πολυβηματικά προβλήματα.

Μείγμα Ειδικών (Mixture of Experts, MoE)

Αντί να ενεργοποιούνται όλα τα δισεκατομμύρια παράμετροι για κάθε λέξη, ένας δρομολογητής επιλέγει δυναμικά μόνο λίγα εξειδικευμένα υποδίκτυα ανά token, επιτυγχάνοντας τεράστια χωρητικότητα γνώσης με χαμηλό κόστος εκτέλεσης.

Εγγενής Πολυτροπικότητα & Μακρύ Πλαίσιο

Ενιαία μοντέλα επεξεργάζονται ταυτόχρονα κείμενο, εικόνες, ήχο, βίντεο και κώδικα σε παράθυρα συμφραζομένων εκατομμυρίων tokens, σε συνδυασμό με τεχνικές ανάκτησης γνώσης (**RAG**) για τεκμηριωμένες απαντήσεις.

Αυτόνομοι Πράκτορες (AI Agents)

Συστήματα που συνδέουν γλωσσικά μοντέλα με εργαλεία (τερματικό, βάσεις δεδομένων, διερμηνείς κώδικα, API, πρωτόκολλο MCP), σχεδιάζοντας, εκτελώντας και επαληθεύοντας αυτόνομα σύνθετες ροές εργασίας υπό ανθρώπινη εποπτεία.

5.4 Εργαστήριο 22: Αυτοπροσοχή Πολλαπλών Κεφαλών σε Καθαρή NumPy

ΕΡΓΑΣΤΗΡΙΟ 22 **Υλοποίηση Scaled Dot-Product & Multi-Head Self-Attention από το Μηδέν σε NumPy**

`code/22_transformer_self_attention.py`

Απομυθοποιεί πλήρως τον πυρήνα των Transformers υλοποιώντας σε καθαρή **NumPy** την αριθμητικά ευσταθή `softmax` (με αφαίρεση μεγίστου), την εξίσωση `scaled_dot_product_attention(Q, K, V)` και την παράλληλη `multi_head_attention(X, num_heads=4)` πάνω σε ελληνική πρόταση 6 tokens ($d_{model} = 16 \Rightarrow d_k = 4$ ανά κεφαλή), εκτυπώνοντας τον ακριβή πίνακα πιθανοτήτων προσοχής.

Προτεινόμενος Πειραματισμός: Αφαιρέστε προσωρινά τη διαίρεση με `np.sqrt(float(d_k))` και αυξήστε το `d_model` από 16 σε 512: παρατηρήστε πώς η κατανομή Softmax καταρρέει σε ακραία 0 και 1 (κορεσμός κλίσης), επαληθεύοντας εμπειρικά τη θεωρητική απόδειξη της Ενότητας 5.1.

PYTHON 3.12+ `code/22_transformer_self_attention.py` Υλοποίηση Scaled Dot-Product & Multi-Head Self-Attention

```
import numpy as np
```

```python
# === 1. Αριθμητικά Ευσταθής Softmax & Βαθμωτή Αυτοπροσοχή ===

def softmax(x: np.ndarray, axis: int = -1) -> np.ndarray:
    """Αριθμητικά ευσταθής συνάρτηση Softmax με αφαίρεση μεγίστου κατά μήκος του άξονα."""
    shifted = x - np.max(x, axis=axis, keepdims=True)
    exp_vals = np.exp(shifted)
    return exp_vals / np.sum(exp_vals, axis=axis, keepdims=True)


def scaled_dot_product_attention(
    Q: np.ndarray, K: np.ndarray, V: np.ndarray
) -> tuple[np.ndarray, np.ndarray]:
    """
    Υπολογίζει την κλιμακωτή αυτοπροσοχή εσωτερικού γινομένου:
        scores = (Q @ K^T) / sqrt(d_k)
        weights = Softmax(scores)
        output = weights @ V
    Η διαίρεση με sqrt(d_k) αποτρέπει τον κορεσμό της Softmax όταν η διάσταση d_k μεγαλώνει.
    """
    d_k = Q.shape[-1]
    scores = (Q @ K.T) / np.sqrt(float(d_k))
    weights = softmax(scores, axis=-1)
    context = weights @ V
    return context, weights


# === 2. Μηχανισμός Αυτοπροσοχής Πολλαπλών Κεφαλών (Multi-Head Self-Attention) ===

def multi_head_attention(
    X: np.ndarray, num_heads: int, rng: np.random.Generator
) -> tuple[np.ndarray, list[np.ndarray]]:
    """Εφαρμόζει Multi-Head Self-Attention χωρίζοντας τον χώρο ενσωμάτωσης σε παράλληλες κεφαλές."""
    seq_len, d_model = X.shape
    if d_model % num_heads != 0:
        raise ValueError("Η διάσταση d_model πρέπει να διαιρείται ακριβώς με το num_heads.")

    d_head = d_model // num_heads
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/22_transformer_self_attention.py του code.zip)
```

ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
Μήκος ακολουθίας: 6 tokens | d_model = 16 | Κεφαλές = 4 | Έξοδος: (6, 16)
νομισούνη | 0.133  0.133  0.198  0.309  0.176  0.051  (Σ = 1.000)
μαθαίνει  | 0.111  0.131  0.185  0.273  0.221  0.078  (Σ = 1.000)
```

Χαρακτηριστικό	Αναδρομικά Δίκτυα (RNN / LSTM)	Αρχιτεκτονική Transformers
Τρόπος Επεξεργασίας	Σειριακή ($O(N)$ διαδοχικά βήματα στον χρόνο)	Πλήρως παράλληλη σε ολόκληρη την ακολουθία ($O(1)$ βάθος)
Εξαρτήσεις Μεγάλης Απόστασης	Εξασθενούν σταδιακά καθώς αυξάνεται το μήκος N	Άμεση σύνδεση $O(1)$ μεταξύ οποιωνδήποτε δύο tokens (κόστος $O(N^2)$)
Αξιοποίηση GPU / Κλιμάκωση	Περιορισμένη παραλληλοποίηση κατά την εκπαίδευση	Εξαιρετική κλιμάκωση σε τρισεκατομμύρια tokens & Tensor Cores

◈ Κεντρικό Συμπέρασμα Κεφαλαίου 5

Κάθε γραμμή του πίνακα προσοχής αθροίζει ακριβώς στο **1.000** (κατανομή πιθανότητας Softmax), εκφράζοντας πόσο «βάρος» αποδίδει κάθε λέξη της πρότασης στις υπόλοιπες. Σε συνδυασμό με την κρυφή μνήμη **KV-Cache** και τις περιστροφικές κωδικοποιήσεις θέσης (**RoPE**), αυτός ο μηχανισμός αποτελεί τη θεμελιώδη δομική μονάδα όλων των σύγχρονων μοντέλων γλώσσας, συλλογισμού και πολυτροπικής κατανόησης (αναλύονται διεξοδικά στα Κεφάλαια 9 έως 12).



ΚΕΦΑΛΑΙΟ 6

Αξιολόγηση, Γενίκευση και Κύκλος Ζωής Μοντέλων

Αποσύνθεση Μεροληψίας-Διακύμανσης (Bias-Variance), κανονικοποίηση L1/L2, πλήρης οδηγός 10 μετρικών ταξινόμησης και παλινδρόμησης, K-Fold έναντι χρονικής επικύρωσης Walk-Forward και Εργαστήρια 09 και 10.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Κατανόηση του επαναληπτικού κύκλου ζωής ενός συστήματος ML και του αυστηρού διαχωρισμού **Training / Validation / Test**.
- Μαθηματική **Αποσύνθεση Μεροληψίας-Διακύμανσης (Bias-Variance Decomposition)** και σύνδεσή της με υποεκπαίδευση, υπερεκπαίδευση, **L1/L2 Regularization** και **EarlyStopping**.
- Πλήρης έλεγχος όλων των κύριων μετρικών αξιολόγησης (**Accuracy, Precision, Recall, F1, ROC-AUC, PR-AUC, MAE, MSE, RMSE, R^2**) και διάκριση **K-Fold CV** έναντι **Χρονικής Επικύρωσης Walk-Forward**.
- Εκτέλεση των **Εργαστηρίων 09 & 10** για διαφανή υπολογισμό μετρικών σε Pandas DataFrame και επαλήθευση της στοχαστικής σύγκλισης Monte Carlo $O(N^{-1/2})$.

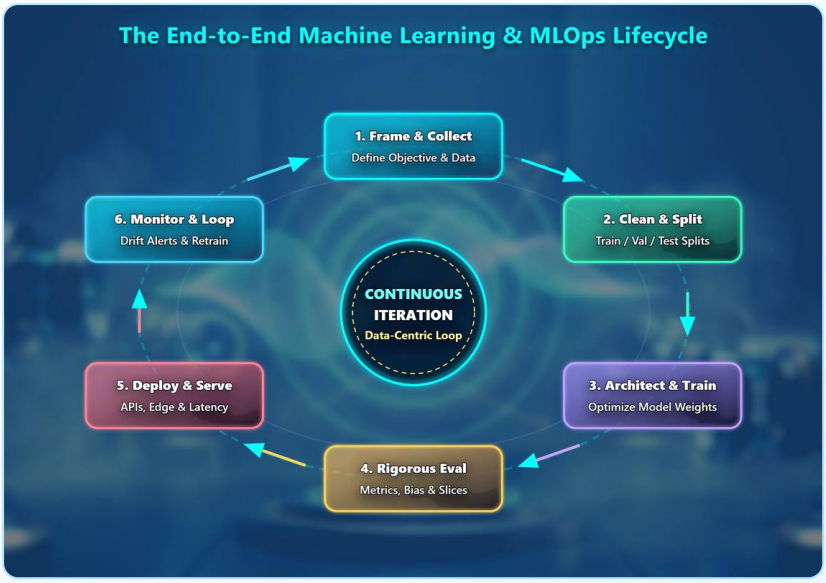
6.1 Ο Ολοκληρωμένος Κύκλος Ζωής ενός Έργου Μηχανικής Μάθησης

Η αξιοπιστία ενός συστήματος AI δεν καθορίζεται μόνο από την επιλογή αρχιτεκτονικής, αλλά από την επιστημονική πειθαρχία σε ολόκληρη τη ροή εργασίας: σαφής διατύπωση προβλήματος, καθαρισμός και ανάλυση δεδομένων, αυστηρή απομόνωση σε **Σύνολο Εκπαίδευσης (Training)**, **Επικύρωσης (Validation)** για επιλογή υπερπαραμέτρων και **Τελικής Αξιολόγησης (Test)** που αγγίζεται *μία μόνο φορά* στο τέλος, ακολουθούμενη από συνεχή παρακολούθηση μετατόπισης κατανομών (*data drift*) σε παραγωγικό περιβάλλον.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ **Η Αναλογία των Πανεπιστημιακών Εξετάσεων: Training, Validation & Test Set**

Γιατί δεν μετράμε ποτέ την επιτυχία ενός μοντέλου πάνω στα ίδια δεδομένα με τα οποία εκπαιδεύτηκε; Σκεφτείτε την προετοιμασία σας για τις εξετάσεις του εξαμήνου:

- **Σύνολο Εκπαίδευσης (Training Set):** Οι ασκήσεις που λύνετε στο σπίτι για να μάθετε τη θεωρία.
- **Σύνολο Επικύρωσης (Validation Set):** Τα δοκιμαστικά διαγωνίσματα προσομοίωσης που σας δείχνουν αν χρειάζεστε αλλαγή μεθόδου (ρύθμιση *υπερπαραμέτρων*).
- **Σύνολο Τελικής Αξιολόγησης (Test Set):** Ο σφραγισμένος φάκελος της τελικής εξέτασης που ανοίγεται *μία μόνο φορά* στο τέλος!
- **Υπερεκπαίδευση (Overfitting) vs Υποεκπαίδευση (Underfitting):** Η αποστήθιση παλιών λύσεων χωρίς κατανόηση δίνει 100% εξάσκηση και αποτυχία στις εξετάσεις (Overfitting), ενώ η ελλιπής μελέτη αποτυγχάνει παντού (Underfitting).
- **Διαρροή Δεδομένων (Data Leakage):** Σαν να κρυφοκοιτάζατε τα θέματα των εξετάσεων ενώ διαβάζατε: ο βαθμός φαίνεται άριστος στο χαρτί, αλλά στον πραγματικό κόσμο το μοντέλο καταρρέει!



Σχήμα 6.1

Ο πλήρης επαναληπτικός κύκλος ζωής ενός μοντέλου Μηχανικής Μάθησης (δεδομένα, εκπαίδευση, επικύρωση, διάθεση, παρακολούθηση).

6.2 Ανάλυση Μεροληψίας-Διακύμανσης (Bias-Variance), Κανονικοποίηση & Επικύρωση

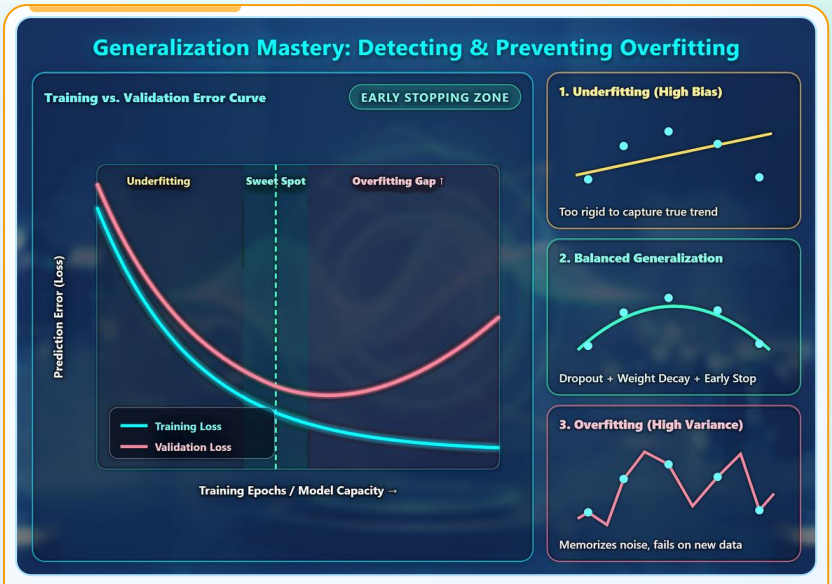
Φανταστείτε έναν τοξότη που στοχεύει το κέντρο ενός στόχου. Εάν όλα τα βέλη του συγκεντρώνονται σφιχτά αλλά συστηματικά εκτός κέντρου, έχει υψηλή **Μεροληψία (Bias)**: το μοντέλο είναι υπερβολικά απλό για να αποτυπώσει την αληθινή φυσική σχέση (**Υποεκπαίδευση / Underfitting**). Αντίθετα, εάν τα βέλη διασκορπίζονται χαοτικά γύρω από το κέντρο ανάλογα με την παραμικρή ριπή ανέμου, έχει υψηλή **Διακύμανση (Variance)**: το μοντέλο απομνημονεύει τον τυχαίο θόρυβο του συγκεκριμένου δείγματος εκπαίδευσης και χάνει τη γενίκευσή του (**Υπερεκπαίδευση / Overfitting**). Μαθηματικά, το αναμενόμενο τετραγωνικό σφάλμα αποσυντίθεται ακριβώς σε τρεις ανεξάρτητους όρους:

$$E[\,(y - \hat{y}(x))^2\,] = [\,\mathrm{Bias}(\hat{y}(x))\,]^2 + \mathrm{Var}(\hat{y}(x)) + \sigma^2$$

$$\textbf{Κανονικοποίηση L1 (Lasso):}\ \ L + \lambda\Sigma|w_j| \quad | \quad \textbf{Κανονικοποίηση L2 (Ridge / Weight Decay):}\ \ L + \lambda\Sigma w_j^2$$

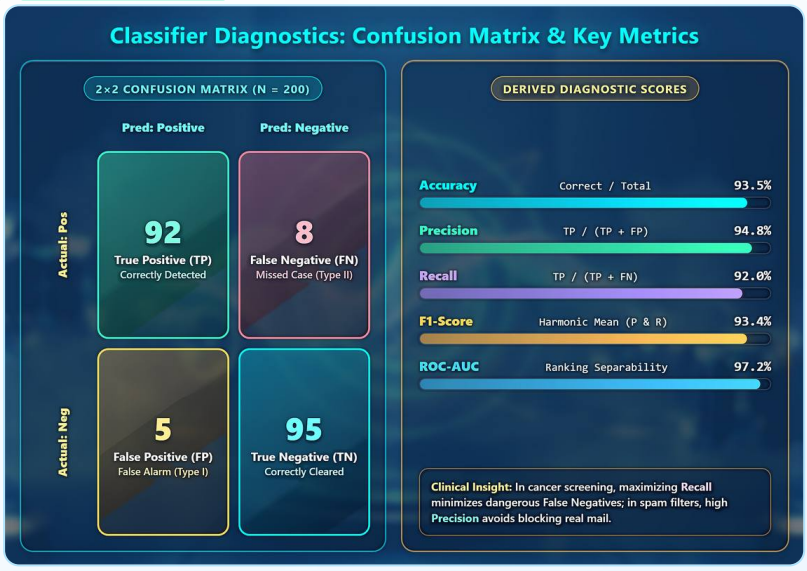
Όπου σ^2 ο μη αναγώγιμος θόρυβος μέτρησης (θεωρητικό κάτω όριο σφάλματος). Η L1 ωθεί ασήμαντα βάρη ακριβώς στο μηδέν (αραιότητα / επιλογή χαρακτηριστικών), ενώ η L2 συρρικνώνει ομαλά όλα τα μεγάλα βάρη περιορίζοντας τη διακύμανση.

Πέρα από την ρητή κανονικοποίηση **L1 / L2** και το **Dropout**, η **Πρόωρη Διακοπή (EarlyStopping)** δρα ως σιωπηρή κανονικοποίηση, τερματίζοντας την εκπαίδευση στο βέλτιστο ελάχιστο της καμπύλης U επικύρωσης (*Optimal Sweet Spot*) πριν η απώλεια επικύρωσης αρχίσει να αυξάνεται. Κρίσιμη σημασία έχει επίσης το πρωτόκολλο επικύρωσης: για ανεξάρτητα και ισοκατανεμημένα δεδομένα (*i.i.d.*, π.χ. πίνακες, εικόνες) εφαρμόζουμε **K-Fold Cross-Validation** (5 ή 10 τμήματα), ενώ σε **χρονοσειρές** (μετεωρολογία, χρηματοοικονομικά, αισθητήρες) απαγορεύεται αυστηρά η τυχαία ανάμειξη! Επιβάλλεται **Χρονική Επικύρωση Προοδευτικού Βήματος (Walk-Forward Split)** και προσαρμογή κάθε κανονικοποιητή (`MinMaxScaler / StandardScaler`) *αποκλειστικά* στο παρελθόν σύνολο εκπαίδευσης, ώστε να αποτρέπεται η καταστροφική **διαρροή μελλοντικής πληροφορίας (future data leakage)**.



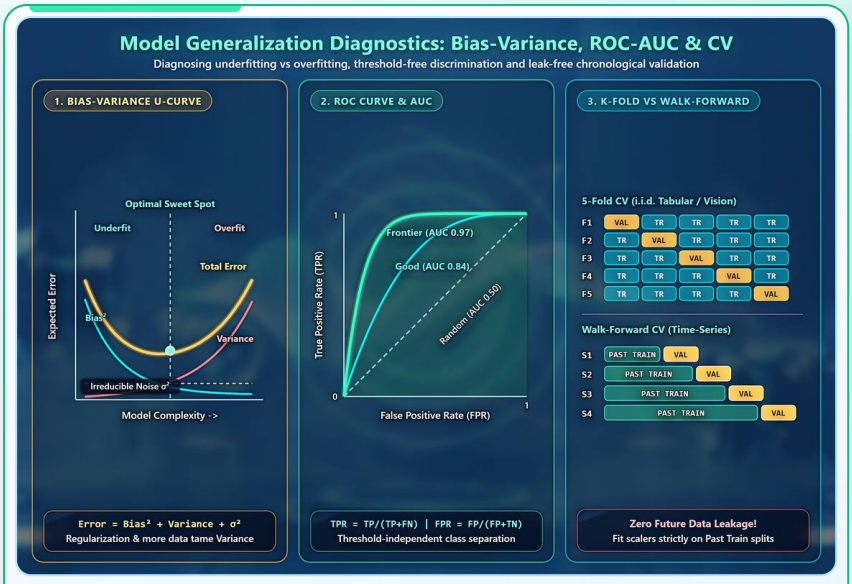
Σχήμα 6.2α

Σύγκριση Υποεκπαίδευσης (Underfitting), Βέλτιστης Γενίκευσης και Υπερεκπαίδευσης (Overfitting).



Σχήμα 6.2β

Πίνακας Σύγχυσης (TP, TN, FP, FN) και παραγωγή των μετρικών Accuracy, Precision, Recall, F1-Score και ROC-AUC.



Σχήμα 6.3

Διαγνωστική τριάδα γενίκευσης: καμπύλη U Μεροληψίας-Διακύμανσης (Bias-Variance), καμπύλη ROC-AUC ανεξάρτητη από κατώφλι και σύγκριση 5-Fold CV έναντι Χρονικής Επικύρωσης Walk-Forward.

6.3 Συνολικός Οδηγός Μετρικών Ταξινόμησης & Παλινδρόμησης

Η επιλογή ακατάλληλης μετρικής μπορεί να κρύψει σοβαρές αστοχίες: σε ένα ιατρικό ή ραδιοαστρονομικό σύνολο όπου το 99% των δειγμάτων είναι αρνητικά, ένας τετριμμένος ταξινομητής που προβλέπει μόνιμα «0» πετυχαίνει 99% *Accuracy* ενώ έχει 0% *Recall*! Ο παρακάτω πίνακας συνοψίζει τις δέκα θεμελιώδεις μετρικές αναφοράς:

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Το Παράδοξο της Ακρίβειας: Ιατρικός Έλεγχος σε 100 Ασθενείς (10 Ασθενείς, 90 Υγιείς)

Φανταστείτε ένα διαγνωστικό τεστ σε **100 εξεταζόμενους**, εκ των οποίων οι **10 είναι πραγματικά ασθενείς** (Θετική κλάση) και οι **90 είναι υγιείς** (Αρνητική κλάση). Συγκρίνουμε δύο συστήματα:

- **Μοντέλο Α (Αφελής ταξινομητής «Όλοι Υγιείς»):** Προβλέπει μόνιμα «Υγιής» για όλους (*TP = 0, FN = 10, FP = 0, TN = 90*). Πετυχαίνει φαινομενικά εξαιρετική **Accuracy = (0 + 90) / 100 = 90,0%**, αλλά έχει **Recall = 0 / (0 + 10) = 0,0%**: έχασε και τους 10 ασθενείς, αποτέλεσμα ιατρικά καταστροφικό!
- **Μοντέλο Β (Πραγματικός ανιχνευτής):** Εντοπίζει 9 από τους 10 ασθενείς (*TP = 9*), χάνει 1 (*FN = 1*), δίνει 5 ψευδείς συναγερμούς σε υγιείς (*FP = 5*) και καθησυχάζει σωστά τους υπόλοιπους 85 (*TN = 85*):
  - **Accuracy** = *(9 + 85) / 100* = ***94,0%*** | **Precision** = *TP / (TP + FP) = 9 / (9 + 5) = 9 / 14* = ***64,3%***
  - **Recall** = *TP / (TP + FN) = 9 / (9 + 1) = 9 / 10* = ***90,0%*** | **F1-Score** = *2 · P · R / (P + R) = 18 / 24* = ***75,0%***.

Χρυσός κανόνας: Όταν η κλάση που αναζητούμε είναι σπάνια (ιατρική διάγνωση, απάτη, σήματα SETI), δεν εμπιστευόμαστε ποτέ μόνη την `Accuracy`: ελέγχουμε πάντα **Recall, Precision, F1 και PR-AUC**.

Μετρική	Τύπος / Ορισμός	Κατηγορία	Πότε Επιλέγεται & Ερμηνεία
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Ταξινόμηση	Αξιόπιστη μόνο σε ισορροπημένες κλάσεις. Παραπλανητική σε σπάνια συμβάντα.
Precision	TP / (TP + FP)	Ταξινόμηση	Κρίσιμη όταν οι ψευδείς συναγερμοί (FP) κοστίζουν ακριβά (π.χ. φιλτράρισμα spam, τηλεσκοπικές ειδοποιήσεις).
Recall (TPR)	TP / (TP + FN)	Ταξινόμηση	Κρίσιμη όταν η απώλεια αληθινού περιστατικού (FN) είναι επικίνδυνη (π.χ. ογκολογικός έλεγχος, ανίχνευση SETI).
F1-Score	2 · Precision · Recall / (P + R)	Ταξινόμηση	Αρμονικός μέσος P και R: τιμωρεί αυστηρά όποια από τις δύο μετρικές πέφτει χαμηλά.

Μετρική	Τύπος / Ορισμός	Κατηγορία	Πότε Επιλέγεται & Ερμηνεία
ROC-AUC	Εμβαδόν καμπύλης TPR έναντι FPR	Ταξινόμηση	Αξιολογεί τη διαχωριστική ισχύ σε όλα τα κατώφλια πιθανότητας (0.50 τυχαίο, 1.00 τέλειο).
PR-AUC	Εμβαδόν καμπύλης Precision έναντι Recall	Ταξινόμηση	Χρυσό πρότυπο για ακραία ανισόρροπες κλάσεις, καθώς εστιάζει αποκλειστικά στη σπάνια θετική κλάση.
MAE	$(1/N)\ \Sigma\ \lvert y_i - \hat{y}_i \rvert$	Παλινδρόμηση	Γραμμικό σφάλμα στις φυσικές μονάδες, ανθεκτικό σε ακραίες τιμές (*outliers*).
MSE	$(1/N)\ \Sigma\ (y_i - \hat{y}_i)^2$	Παλινδρόμηση	Ομαλά διαφορίσιμη απώλεια που τιμωρεί τετραγωνικά τα μεγάλα σφάλματα πρόβλεψης.
RMSE	$\sqrt{[\ (1/N)\ \Sigma\ (y_i - \hat{y}_i)^2\ ]}$	Παλινδρόμηση	Αποδίδει το τετραγωνικό σφάλμα στις φυσικές μονάδες του μεγέθους (π.χ. °C, hPa).
R^2	$1 - \Sigma(y_i - \hat{y}_i)^2 / \Sigma(y_i - \bar{y})^2$	Παλινδρόμηση	Ποσοστό διακύμανσης που εξηγεί το μοντέλο έναντι του απλού μέσου όρου (1.0 τέλειο, <0 χειρότερο από μέσο).

6.4 Εργαστήρια 09 & 10: Μετρικές Αξιολόγησης & Στοχαστική Σύγκλιση Monte Carlo

ΕΡΓΑΣΤΗΡΙΟ 09

Υπολογισμός Μετρικών Παλινδρόμησης (MSE, RMSE, MAE, R2), Πίνακα Σύγχυσης & Pandas DataFrame

`code/09_metrics_rmse_and_dataframe.py`

Υλοποιεί από το μηδέν με καθαρή **NumPy** τόσο τις μετρικές παλινδρόμησης (`MSE, RMSE, MAE, R2`) όσο και τον πλήρη Πίνακα Σύγχυσης (`TP, TN, FP, FN → Accuracy, Precision, Recall, F1`) με ασφαλή προστασία από διαίρεση με το μηδέν, και συγκεντρώνει τα αποτελέσματα σε οργανωμένο **Pandas DataFrame**.

Προτεινόμενος Πειραματισμός: Προσθέστε μία τεχνητή ακραία τιμή (*outlier*) στις προβλέψεις παλινδρόμησης και παρατηρήστε πώς το **RMSE** εκτοξεύεται πολύ εντονότερα από το **MAE**, επιβεβαιώνοντας τη διαφορετική ευαισθησία τους στα ακραία σφάλματα.

PYTHON 3.12+ `code/09_metrics_rmse_and_dataframe.py` — Υπολογισμός MSE, RMSE, MAE, R2 & Πίνακα Σύγχυσης

```python
import numpy as np

try:
    import pandas as pd
except ImportError:
    pd = None


def compute_regression_metrics(y_true: np.ndarray, y_pred: np.ndarray) -> dict[str, float]:
    """Υπολογίζει διαφανώς τις κύριες μετρικές σφάλματος παλινδρόμησης (MSE, RMSE, MAE, R^2)."""
    errors = y_true - y_pred
    mse = float(np.mean(errors ** 2))
    rmse = float(np.sqrt(mse))
    mae = float(np.mean(np.abs(errors)))

    # Υπολογισμός συντελεστή προσδιορισμού R^2 = 1 - (SS_res / SS_tot)
    ss_res = float(np.sum(errors ** 2))
    ss_tot = float(np.sum((y_true - np.mean(y_true)) ** 2))
    r2 = 1.0 - (ss_res / ss_tot) if ss_tot > 0 else 0.0
    return {"MSE": mse, "RMSE": rmse, "MAE": mae, "R2": r2}


def compute_binary_classification_metrics(
    y_true: np.ndarray, y_pred: np.ndarray
) -> dict[str, float]:
    """Υπολογίζει τα στοιχεία του Πίνακα Σύγχυσης (TP, TN, FP, FN) και Accuracy, Precision, Recall, F1."""
```

```
    # Καταμέτρηση αληθώς/ψευδώς θετικών και αρνητικών προβλέψεων
    tp = int(np.sum((y_true == 1) & (y_pred == 1)))
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/09_metrics_rmse_and_dataframe.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 10

Γεωμετρική Πιθανότητα Monte Carlo (π), Σύγκλιση O(N^-1/2) & Σύνδεση με Νευρωνική Παλινδρόμηση Ορίου

`code/10_calc_pi_monte_carlo_nn.py`

Εφαρμόζει διανυσματική στοχαστική δειγματοληψία **Monte Carlo** στο μοναδιαίο τετράγωνο $[0, 1]^2$ έως **5.000.000 δείγματα**: ο λόγος σημείων που ικανοποιούν το κυκλικό όριο $x^2 + y^2 \leq 1$ συγκλίνει στο $\pi/4$. Κατά το Κεντρικό Οριακό Θεώρημα, η τυπική απόκλιση της εκτίμησης μειώνεται ως $O(N^{-1/2})$, νόμος που επαληθεύεται εμπειρικά με λογαριθμική παλινδρόμηση (`np.polyfit`). Στη βαθιά μάθηση, η ίδια μη γραμμική κυκλική συνάρτηση δείκτη $I(x^2 + y^2 \leq 1)$ αποτελεί κλασικό πρόβλημα αναφοράς **νευρωνικής παλινδρόμησης ορίου** για τη μέτρηση της σύγκλισης γενίκευσης καθώς αυξάνεται το μέγεθος του εκπαιδευτικού δείγματος N.

Προτεινόμενος Πειραματισμός: Δοκιμάστε να αυξήσετε το μέγιστο N στα 10.000.000 δείγματα και επαληθεύστε τον θεμελιώδη κανόνα κόστους των μεθόδων Monte Carlo: για κάθε **1 επιπλέον δεκαδικό ψηφίο ακρίβειας** (10× μικρότερο σφάλμα) απαιτούνται **100× περισσότερα δείγματα**!

PYTHON 3.12+ `code/10_calc_pi_monte_carlo_nn.py` Διανυσματική εκτίμηση π (Monte Carlo) & επαλήθευση log-log σύγκλισης

```python
import math
import numpy as np


def estimate_pi_monte_carlo(num_samples: int, rng: np.random.Generator) -> float:
    """Εκτιμά την τιμή του π από num_samples ομοιόμορφα τυχαία σημεία στο [0, 1] x [0, 1]."""
    # Διανυσματική παραγωγή συντεταγμένων (x, y) με NumPy για υψηλή υπολογιστική απόδοση
    x = rng.random(num_samples, dtype=np.float64)
    y = rng.random(num_samples, dtype=np.float64)

    # Έλεγχος εμπτώσεως εντός του μοναδιαίου τεταρτημορίου: x^2 + y^2 <= 1
    inside_quarter_circle = np.sum((x * x + y * y) <= 1.0)

    # Ο λόγος εμβαδών (τεταρτημόριο κύκλου / μοναδιαίο τετράγωνο) ισούται με π / 4
    return 4.0 * float(inside_quarter_circle) / float(num_samples)


def main() -> None:
    """Εκτελεί πειράματα Monte Carlo σε κλιμακούμενο N και επαληθεύει τον ρυθμό σύγκλισης."""
    # === 1. Ορισμός Μεγεθών Δείγματος & Εκτέλεση Προσομοιώσεων Monte Carlo ===
    rng = np.random.default_rng(seed=42)
    sample_sizes = [1_000, 5_000, 10_000, 50_000, 100_000, 500_000, 1_000_000, 5_000_000]

    print(f"Πραγματική τιμή π (math.pi): {math.pi:.10f}\n")
    print(f"{'Δείγματα N':>12s} | {'Εκτίμηση π':>14s} | {'Απόλυτο Σφάλμα':>16s}")
    print("-" * 50)

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/10_calc_pi_monte_carlo_nn.py του code.zip)
```

ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
N = 1,000 -> π ≈ 3.1496 (σφάλμα 0.0413)  |  N = 5,000,000 -> π ≈ 3.1422 (σφάλμα 0.00084)
Εμπειρική κλίση log-log: -0.4231 (θεωρητική τιμή: -0.5000, νόμος O(N^-1/2))
```

ΚΕΦΑΛΑΙΟ 7

Οικοσύστημα Λογισμικού, Οπτικοποίηση & Επιθεώρηση Μοντέλων

Συγκριτική αρχιτεκτονική της πλήρους στοίβας μηχανικής AI (NumPy έως ONNX, TensorBoard, Netron, H5Web), συστηματικό πρωτόκολλο αποσφαλμάτωσης μοντέλων και Εργαστήριο 04.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Χαρτογράφηση ολόκληρης της **Στοίβας Μηχανικής AI (AI Engineering Stack)** σε Python 3.12+ από τη διανυσματική άλγεβρα έως τη διαλειτουργική εξαγωγή ONNX.
- Μετατροπή των νευρωνικών δικτύων από αδιαφανή «μαύρα κουτιά» σε πλήρως παρατηρήσιμα επιστημονικά αντικείμενα με **TensorBoard, Netron, H5Web** και **Neuroph Studio**.
- Εφαρμογή συστηματικού **Πρωτοκόλλου Αποσφαλμάτωσης & Παρατηρησιμότητας** (καμπύλες απώλειας, νόρμες κλίσης, ιστογράμματα βαρών, έλεγχος διαστάσεων τανυστών).
- Καταγραφή μετρικών και ιστογραμμάτων βαρών ανά εποχή στο **Εργαστήριο 04** (`04_and_gate_tensorboard.py`).

7.1 Πλήρης Στοίβα Μηχανικής AI (AI Engineering Stack) σε Python 3.12+

Η σύγχρονη έρευνα και ανάπτυξη εφαρμογών βαθιάς μάθησης δεν βασίζεται σε ένα μεμονωμένο εργαλείο, αλλά σε μια διαστρωματωμένη, ανοικτή στοίβα βιβλιοθηκών όπου κάθε επίπεδο εξυπηρετεί διακριτό επιστημονικό και μηχανικό ρόλο:

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Το Εργαστήριο του Μηχανικού AI: Γιατί Υπάρχουν Τόσες Βιβλιοθήκες & Τι Κάνει η Καθεμία

Για έναν νέο φοιτητή, ονόματα όπως *NumPy, Pandas, Keras, PyTorch, TensorBoard, Netron, ONNX* μπορεί αρχικά να φαίνονται πολλά. Σκεφτείτε τα σαν ένα καλοοργανωμένο συνεργείο Formula 1 όπου κάθε εργαλείο έχει σαφή αποστολή: το **εικονικό περιβάλλον (`.venv`)** είναι ο καθαρός πάγκος εργασίας κάθε έργου (ώστε οι βιβλιοθήκες ενός έργου να μη συγκρούονται με ένα άλλο), η **NumPy/Pandas** είναι οι υπολογιστές και τα φύλλα μετρήσεων, τα **Keras 3 & PyTorch 2** συναρμολογούν και εκπαιδεύουν τον κινητήρα (το νευρωνικό δίκτυο), ενώ τα **TensorBoard, Netron & H5Web** είναι η οθόνη τηλεμετρίας και ο αξονικός τομογράφος που μας δείχνουν τι συμβαίνει *μέσα* στο δίκτυο ώστε να μην οδηγούμε ποτέ στα τυφλά!

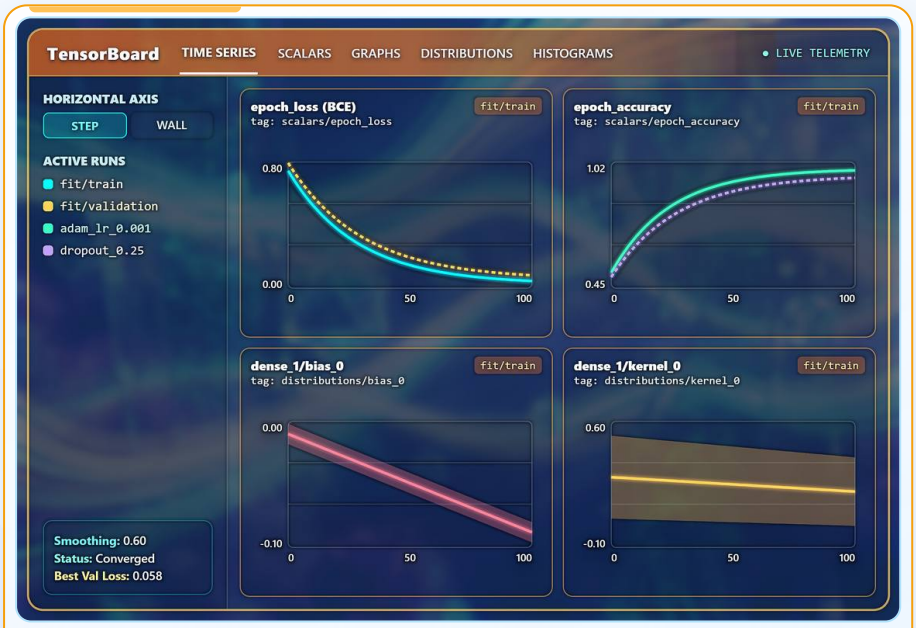
- **API:** Το σύνολο έτοιμων εντολών (π.χ. `model.fit()`) με τις οποίες καλούμε μια βιβλιοθήκη.
- **Σχήμα Τανυστή (Tensor Shape):** Οι διαστάσεις των πινάκων που ρέουν μεταξύ επιπέδων (η νούμερο 1 αιτία σφαλμάτων αρχαρίων).
- **Checkpoint (`.keras / .pt`):** Αποθηκευμένο αρχείο με την αρχιτεκτονική και τα εκπαιδευμένα βάρη για άμεση επαναχρησιμοποίηση.

Εργαλείο / Βιβλιοθήκη	Ρόλος στη Στοίβα AI	Τυπικές Μορφές / Διεπαφές	Κύρια Πλεονεκτήματα & Εφαρμογή στο Σεμινάριο
NumPy 2	Πυρήνας διανυσματικής άλγεβρας	`ndarray` (C/BLAS)	Διαφανής υλοποίηση αλγορίθμων από το μηδέν (Εργαστήρια 02, 06, 09, 10, 22, 23 έως 29).
Pandas 2	Δομημένα δεδομένα & χρονοσειρές	`DataFrame` , CSV, Parquet	Καθαρισμός, χρονική ευθυγράμμιση και συγκριτική πινακοποίηση μετρικών (Εργαστήρια 09, 16, 17).
Keras 3	Πολυ-πυρηνικό DL API υψηλού επιπέδου	`.keras` (TF / PyTorch / JAX)	Καθαρή, ενοποιημένη σύνταξη Sequential & Functional με άμεσο EarlyStopping και TensorBoard.

Εργαλείο / Βιβλιοθήκη	Ρόλος στη Στοίβα AI	Τυπικές Μορφές / Διεπαφές	Κύρια Πλεονεκτήματα & Εφαρμογή στο Σεμινάριο
PyTorch 2	Κυρίαρχη ερευνητική πλατφόρμα DL	`.pt` / `.safetensors`, CUDA	Δυναμικοί γράφοι, `torch.amp`, `torch.compile` και εγγενής GPU σε Windows (Εργ. 11).
Hugging Face	Οικοσύστημα θεμελιωδών μοντέλων	`transformers`, `peft`	Εκατοντάδες χιλιάδες ανοιχτά LLMs, tokenizers, MarianMT μετάφραση (Εργ. 15) και LoRA adapters.
ONNX	Διαλειτουργική εξαγωγή παραγωγής	`.onnx` (ONNX Runtime)	Εκπαίδευση σε PyTorch/Keras και εκτέλεση υψηλής ταχύτητας σε C++, edge συσκευές, TensorRT ή Web.
TensorBoard	Ζωντανή τηλεμετρία εκπαίδευσης	Event logs (`logs/`)	Παρακολούθηση καμπυλών Loss/Accuracy, ιστογραμμάτων βαρών και υπολογιστικού γράφου (Εργ. 04, 17).
Netron	Οπτικός επιθεωρητής τοπολογίας	`.keras, .h5, .onnx, .gguf`	Άμεσος έλεγχος επιπέδων, συνδέσεων Skip και διαστάσεων τανυστών χωρίς εκτέλεση κώδικα.
H5Web	Επιθεώρηση αριθμητικών πινάκων HDF5	`.h5` (VS Code Extension)	Απευθείας προβολή θερμικών χαρτών και αριθμητικών τιμών των συναπτικών βαρών και biases (Εργ. 02).

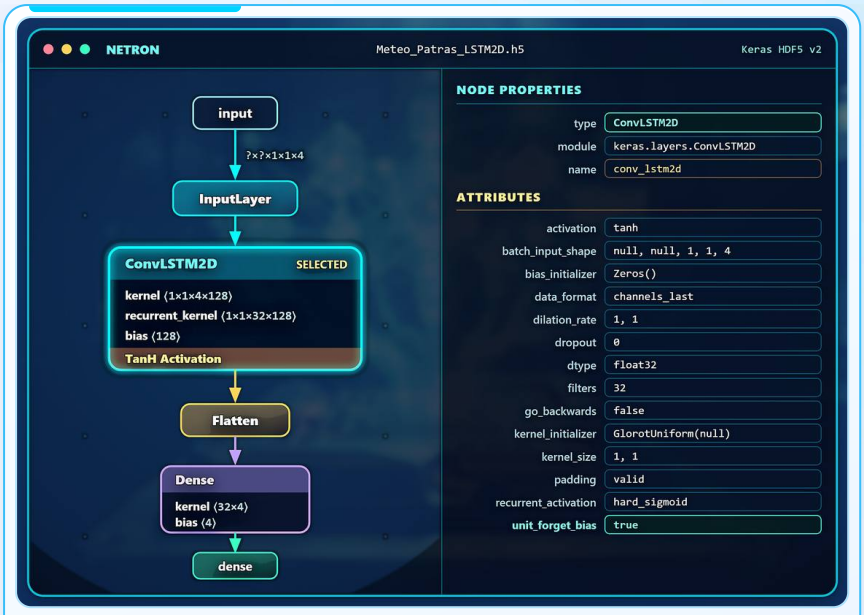
7.2 Οπτικοποίηση, Επιθεώρηση & Συστηματικό Πρωτόκολλο Αποσφαλμάτωσης

Ένα νευρωνικό δίκτυο δεν πρέπει ποτέ να αντιμετωπίζεται ως αδιαφανές «μαύρο κουτί». Τέσσερα εξειδικευμένα οπτικά εργαλεία μετατρέπουν κάθε εκπαίδευση σε μετρήσιμη επιστημονική παρατήρηση:



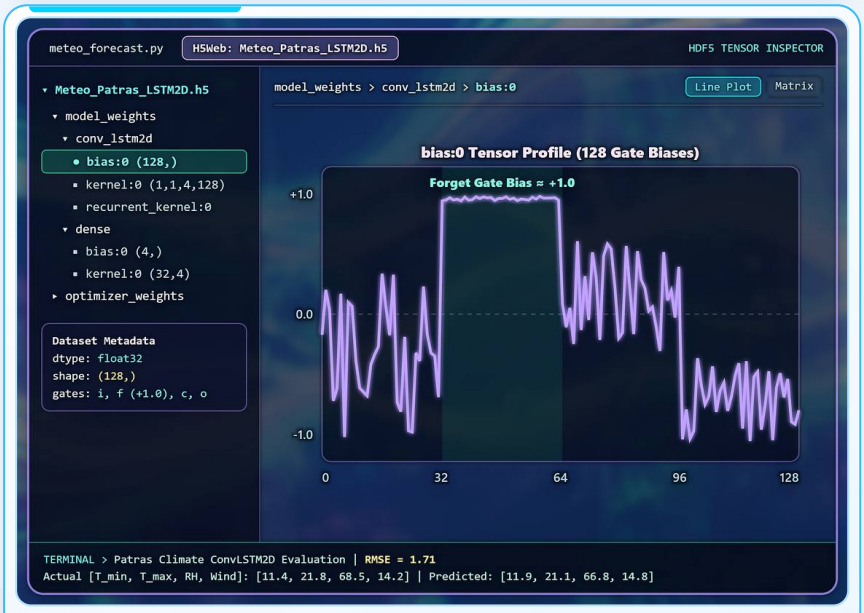
Σχήμα 7.1

TensorBoard: ζωντανή παρακολούθηση καμπυλών Loss/Accuracy και ιστογραμμάτων βαρών ανά εποχή.



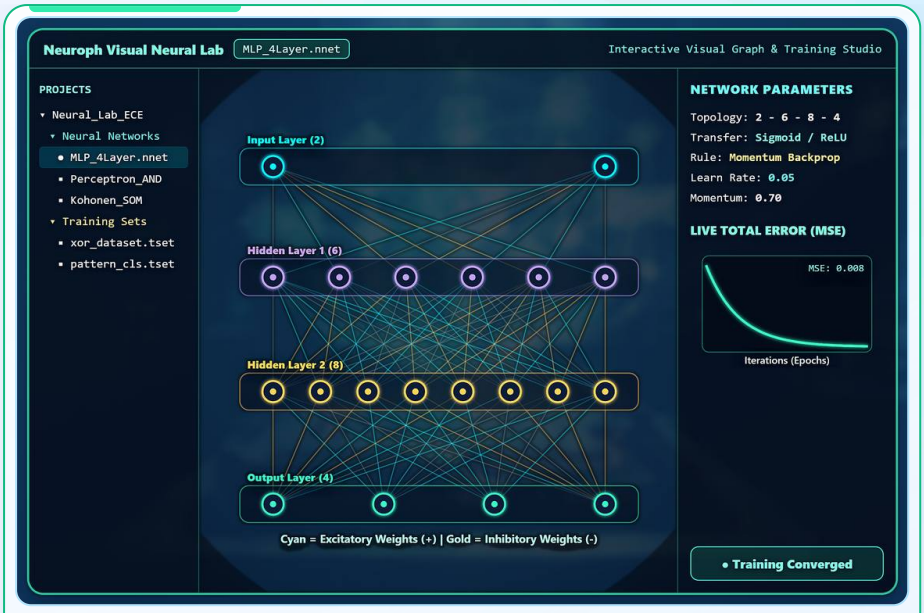
Σχήμα 7.2

Netron: διαδραστική οπτικοποίηση της τοπολογίας, των επιπέδων και των διαστάσεων τανυστών αρχείων .keras, .h5 και .onnx.



Σχήμα 7.3

H5Web (επέκταση Visual Studio Code): άμεση επιθεώρηση πινάκων συναπτικών βαρών και biases μέσα σε αρχεία HDF5 (.h5).



Σχήμα 7.4

Neuroph Studio: εκπαιδευτικό γραφικό περιβάλλον (Java) για άμεσο πειραματισμό με μικρά νευρωνικά δίκτυα και συνάψεις.

◆ Πρωτόκολλο 4 Βημάτων Αποσφαλμάτωσης & Παρατηρησιμότητας Μοντέλων (Model Debugging Protocol)

1. Επαλήθευση Διαστάσεων Τανυστών (Tensor Shapes & Netron): Ελέγξτε πάντα τα σχήματα εισόδου/εξόδου στο `model.summary()` και στο Netron πριν την εκπαίδευση. Σιωπηρά σφάλματα broadcasting (π.χ. `(N,)` έναντι `(N, 1)`) καταστρέφουν την απώλεια χωρίς να εγείρουν εξαίρεση.

2. Διάγνωση Καμπυλών Απώλειας (Loss Forensics): Επίπεδη απώλεια υποδεικνύει πολύ μικρό *learning rate*, λανθασμένη κωδικοποίηση ετικετών ή *Dying ReLUs*. Εκρηκτικές τιμές `NaN` υποδεικνύουν υπερβολικό *lr* ή ακανονικοποίητα χαρακτηριστικά. Απόκλιση `val_loss` προς τα άνω σηματοδοτεί υπερεκπαίδευση. Δοκιμάστε πρώτα σκόπιμη υπερεκπαίδευση σε μία μικρο-δέσμη 4 δειγμάτων (*overfit one batch sanity check*): αν η απώλεια δεν μηδενίζεται, υπάρχει σφάλμα στον κώδικα.

3. Παρακολούθηση Νορμών Κλίσης (Gradient Norms): Καταγράψτε τη νόρμα $\|\nabla_\theta L\|_2$ ανά επίπεδο για έγκαιρο εντοπισμό εξασθένησης ($\to 0$) ή έκρηξης κλίσης (εφαρμογή `clipnorm=1.0`).

4. Ιστογράμματα Βαρών (TensorBoard & H5Web): Υγιή βάρη σχηματίζουν ομαλή κωδωνοειδή κατανομή γύρω από το μηδέν. Αιχμές στα άκρα (±1 ή 0) αποκαλύπτουν κορεσμό ενεργοποιήσεων.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Οδηγός 1ης Ημέρας: Ιχνηλάτηση Σχημάτων Τανυστών & Ανάγνωση model.summary() στο Χέρι

Ας ακολουθήσουμε μια μικρο-δέσμη **32 ασπρόμαυρων εικόνων MNIST** με σχήμα `(32, 28, 28, 1)` μέσα από ένα απλό συνελικτικό δίκτυο, επαληθεύοντας στο χέρι κάθε αριθμό που τυπώνει το `model.summary()`:

- **1.** `Conv2D(32, (3, 3), padding='valid')`**:** Ύψος & πλάτος *28 − 3 + 1 = 26* → σχήμα εξόδου `(32, 26, 26, 32)`. Παράμετροι: *(3 × 3 × 1 + 1 bias) × 32 =* ***320***!
- **2.** `MaxPooling2D((2, 2))`**:** Υποδιπλασιάζει τις χωρικές διαστάσεις (*26 / 2 = 13*) → σχήμα `(32, 13, 13, 32)`. Παράμετροι: **0** (απλή επιλογή μεγίστου).
- **3.** `Flatten()`**:** Ξεδιπλώνει κάθε εικόνα σε 1D διάνυσμα μήκους *13 × 13 × 32 =* ***5.408*** → σχήμα `(32, 5408)`. Παράμετροι: **0**.
- **4.** `Dense(10, activation='softmax')`**:** Παράγει 10 πιθανότητες κλάσεων → σχήμα `(32, 10)`. Παράμετροι: *(5.408 + 1) × 10 =* ***54.090***!

Σύνολο: 54.410 παράμετροι (το 99,4% συγκεντρώνεται στο τελικό `Dense`!). Κάθε φορά που βλέπετε σφάλμα ασυμφωνίας διαστάσεων (`ValueError: incompatible shape`), αυτός ακριβώς ο πίνακας 4 γραμμών εντοπίζει αμέσως το προβληματικό επίπεδο.

7.3 Εργαστήριο 04: Καταγραφή Εκπαίδευσης & Ιστογραμμάτων στο TensorBoard

ΕΡΓΑΣΤΗΡΙΟ 04

Ενσωμάτωση TensorBoard Callback, Χρονοσημασμένων Καταγραφών & Ιστογραμμάτων Βαρών

`code/04_and_gate_tensorboard.py`

Ενσωματώνει το `keras.callbacks.TensorBoard(log_dir=..., histogram_freq=10, write_graph=True)` στην εκπαίδευση του μοντέλου πύλης AND. Κάθε εκτέλεση δημιουργεί μοναδικό χρονοσημασμένο υποκατάλογο `logs/gate_and/run_YYYYMMDD_HHMMSS`, επιτρέποντας τη συγκριτική υπέρθεση πολλαπλών πειραμάτων (π.χ. διαφορετικών *learning rates*) στο ίδιο γράφημα.

Προτεινόμενος Πειραματισμός: Εκτελέστε το πρόγραμμα δύο φορές, πρώτα με `learning_rate=0.04` και έπειτα με `0.005`, και εκκινήστε `tensorboard` στον φάκελο `logs/gate_and` μέσα από το Visual Studio Code για να συγκρίνετε άμεσα τις καμπύλες απώλειας και την εξέλιξη των ιστογραμμάτων βαρών.

PYTHON 3.12+ **code/04_and_gate_tensorboard.py** Ενσωμάτωση TensorBoard Callback & ιστογραμμάτων σε Keras 3

```python
from datetime import datetime
from pathlib import Path
import numpy as np
import keras
from keras import layers


def main() -> None:
    """Εκπαιδεύει μοντέλο AND με ενεργή καταγραφή μετρικών και βαρών στο TensorBoard."""
    keras.utils.set_random_seed(42)

    # === 1. Προετοιμασία Δεδομένων & Αρχιτεκτονικής Μοντέλου ===
    X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
    y = np.array([[0], [0], [0], [1]], dtype=np.float32)

    model = keras.Sequential(
        [
            keras.Input(shape=(2,), name="inputs"),
            layers.Dense(4, activation="relu", name="dense_hidden"),
            layers.Dense(1, activation="sigmoid", name="dense_output"),
        ],
        name="and_gate_tensorboard",
    )

    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=0.04),
        loss="binary_crossentropy",
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/04_and_gate_tensorboard.py του code.zip)
```

ΚΕΦΑΛΑΙΟ 8

Υπολογιστική Επιτάχυνση GPU, CUDA & Περιβάλλον Εργασίας

SIMD έναντι SIMT Warps, Μοντέλο Roofline (Compute-Bound vs Memory-Bound), Συστολικοί Tensor Cores, Αριθμητική Ακρίβεια (FP32 έως FP8), Automatic Mixed Precision (AMP) και Εργαστήρια 11 και 12.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Αρχιτεκτονική σύγκριση **CPU (SIMD / AVX)** έναντι **GPU (SIMT Warps & Συστολικοί Tensor Cores)** και ερμηνεία της πειραματικής επιτάχυνσης στον εργαστηριακό σταθμό.
- Κατανόηση του **Μοντέλου Roofline** (*Αριθμητική Ένταση FLOPs/Byte*) και της διάκρισης μεταξύ πυρήνων **Compute-Bound** και **Memory-Bandwidth-Bound**.
- Πλήρης σύγκριση των προτύπων κινητής υποδιαστολής **FP32, TF32, BF16, FP16, FP8** και λειτουργία του αγωγού **Automatic Mixed Precision (AMP)** με *Dynamic Loss Scaling*.
- Πειραματική μέτρηση επιτάχυνσης και διαχείριση VRAM στα **Εργαστήρια 11 & 12** σε **PyTorch 2** και **Keras 3 / TensorFlow**.

8.1 Αρχιτεκτονική CPU έναντι GPU: SIMD, SIMT Warps, Tensor Cores & Μοντέλο Roofline

Η εκπαίδευση βαθιών νευρωνικών δικτύων κυριαρχείται από μαζικούς πολλαπλασιασμούς πινάκων (**GEMM**) και συνελίξεις τανυστών. Μία **CPU** σχεδιάζεται σαν λίγα υπεραυτοκίνητα αγώνων: διαθέτει ελάχιστους πανίσχυρους πυρήνες, μεγάλες κρυφές μνήμες L1/L2/L3 και σύνθετη πρόβλεψη διακλαδώσεων για *ελάχιστη καθυστέρηση (low latency)* σε σειριακή λογική, επεξεργαζόμενη διανύσματα 8 έως 16 αριθμών ταυτόχρονα μέσω εντολών **SIMD (AVX2 / AVX-512)**. Αντίθετα, μία **GPU** λειτουργεί σαν στόλος χιλιάδων συγχρονισμένων λεωφορείων βελτιστοποιημένων για *μέγιστη διαμεταγωγή (high throughput)*: οργανώνει δεκάδες Streaming Multiprocessors που εκτελούν ομάδες 32 νημάτων ενιαία (**SIMT Warps**), ενώ οι σύγχρονες μονάδες **Tensor Cores** υλοποιούν **συστολικές συστοιχίες (Systolic Arrays)** που εκτελούν την πράξη $D = A \times B + C$ επαναχρησιμοποιώντας κάθε πλακίδιο FP16/BF16 σε δεκάδες πολλαπλασιαστές χωρίς επαναλαμβανόμενες προσπελάσεις στην αργή εξωτερική μνήμη.

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

CPU έναντι GPU, VRAM & Μικτή Ακρίβεια: Η Αναλογία της Τεράστιας Κουζίνας

Γιατί οι κάρτες γραφικών (που φτιάχτηκαν αρχικά για βιντεοπαιχνίδια) έγιναν η καρδιά της Τεχνητής Νοημοσύνης; Φανταστείτε μια **CPU** σαν 8 κορυφαίους σεφ που εκτελούν εξαιρετικά γρήγορα πολύπλοκες, διαδοχικές συνταγές. Αντίθετα, μια **GPU** μοιάζει με μια τεράστια κουζίνα σταδίου με **3.584 βοηθούς (CUDA Cores)** που κόβουν ταυτόχρονα χιλιάδες πανομοιότυπα υλικά στο ίδιο χιλιοστό του δευτερολέπτου! Επειδή η Βαθιά Μάθηση αποτελείται από δισεκατομμύρια απλούς, παράλληλους πολλαπλασιασμούς $w \times x$, η GPU θριαμβεύει. Και η **Μικτή Ακρίβεια (FP16/BF16)** είναι σαν να ζυγίζετε αλεύρι με ακρίβεια γραμμαρίου αντί μικρογραμμαρίου: μισός όγκος μεταφοράς, διπλάσια έως τετραπλάσια ταχύτητα, με πρακτικά ταυτόσημο αποτέλεσμα!

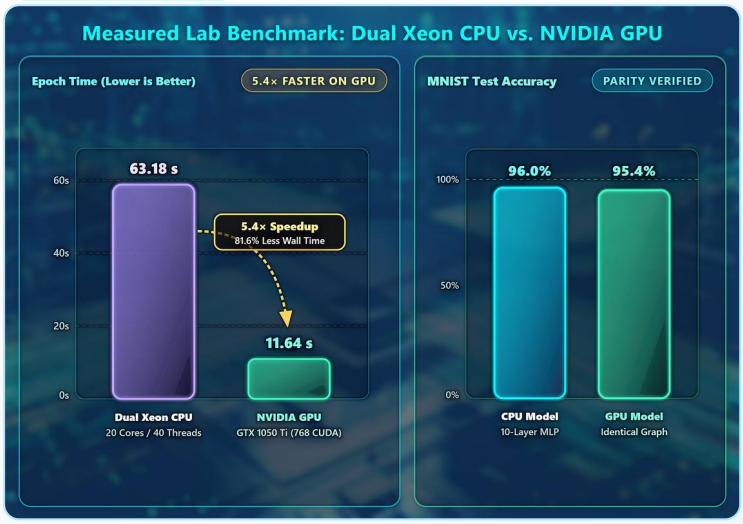
- **FLOP / TFLOPS:** Μία πράξη δεκαδικών αριθμών (1 TFLOPS = 1 τρισεκατομμύριο πράξεις ανά δευτερόλεπτο).
- **VRAM (Video RAM):** Η υπερταχεία μνήμη πάνω στην κάρτα γραφικών (π.χ. 12 GB στην RTX 3060) που χωρά το μοντέλο και το batch.
- **CUDA Out of Memory (OOM):** Όταν μοντέλο και ενεργοποιήσεις ξεπεράσουν τη VRAM: λύνεται μειώνοντας το `batch_size` ή ενεργοποιώντας `AMP (FP16/BF16)`.

Στον εργαστηριακό σταθμό του σεμιναρίου, δύο επεξεργαστές **Intel Xeon E5-2600 v3** (20 φυσικοί πυρήνες, 40 νήματα, 32 GB DDR4) χρειάζονται 63.18 s ανά εποχή στο MNIST, ενώ ακόμη και μία συμπαγής **NVIDIA GeForce GTX 1050 Ti** (768 CUDA Cores, 4 GB GDDR5) ολοκληρώνει την ίδια εποχή σε **11.64 s** (επιτάχυνση **5.4×** με ισοδύναμη ακρίβεια

~96%). Στις σύγχρονες GPU αρχιτεκτονικής RTX / Hopper / Blackwell, οι Tensor Cores φθάνουν εκατοντάδες έως χιλιάδες **TFLOPS**, ξεπερνώντας την 20-πύρηνη CPU κατά περισσότερο από **100×** σε πυκνούς πολλαπλασιασμούς πινάκων!

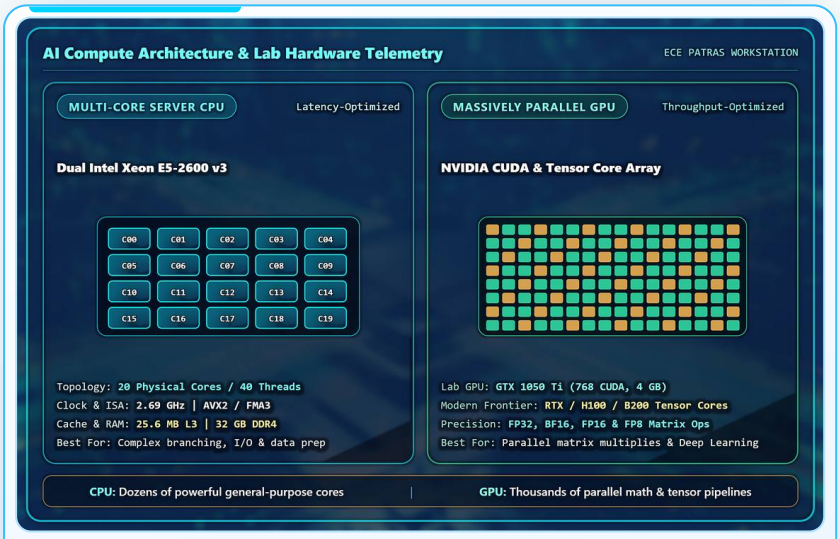
Αριθμητική Ένταση: $I = FLOPs / Bytes_{mem}$ | **Μοντέλο Roofline:** $P_{attainable} = \min(P_{peak}, I \cdot BW_{mem})$

Όταν η αριθμητική ένταση I είναι υψηλή (μεγάλα GEMM, πυκνές Conv2D), ο πυρήνας είναι Compute-Bound (περιορίζεται από τα TFLOPS). Όταν I είναι χαμηλή (ReLU, Softmax, LayerNorm, παραγωγή tokens σε LLM), είναι Memory-Bandwidth-Bound (περιορίζεται από τα GB/s της VRAM), εξηγώντας γιατί η συγχώνευση τελεστών (FlashAttention / torch.compile) επιφέρει τεράστια κέρδη.



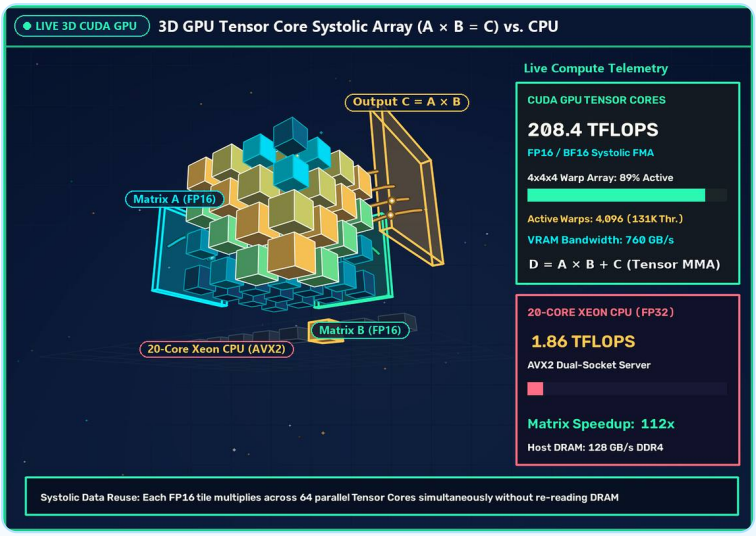
Σχήμα 8.1

Πειραματική σύγκριση χρόνου εποχής (63.18 s σε Dual Xeon CPU έναντι 11.64 s σε NVIDIA GPU) και ισοδυναμίας ακρίβειας.



Σχήμα 8.2

Τηλεμετρία υλικού εργαστηρίου: αρχιτεκτονική Dual Xeon CPU (20C/40T) έναντι συστοιχίας CUDA & Tensor Cores GPU.



Σχήμα 8.3

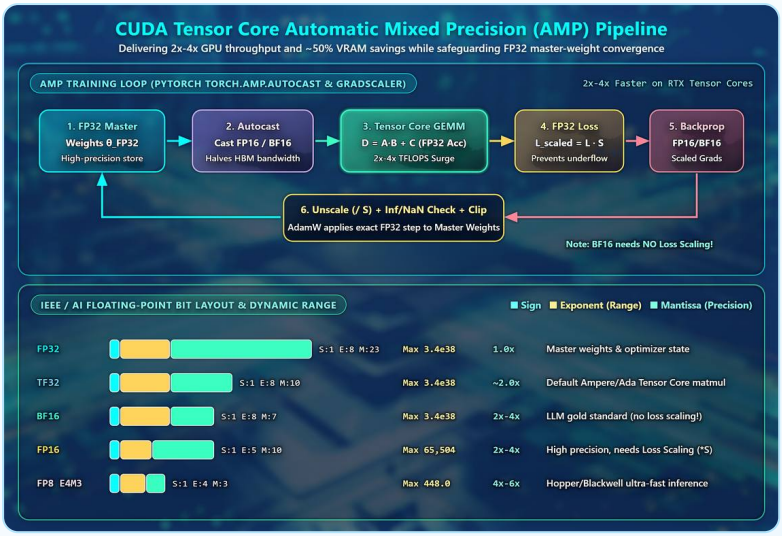
Τρισδιάστατη συστολική συστοιχία GPU Tensor Cores (D = A x B + C) με επαναχρησιμοποίηση δεδομένων FP16 (208.4 TFLOPS, 760 GB/s) έναντι 20-πύρηνης CPU Xeon AVX2 (1.86 TFLOPS).

8.2 Αριθμητική Ακρίβεια (FP32, TF32, BF16, FP16, FP8) & Αυτόματη Μικτή Ακρίβεια (AMP)

Ενώ η κλασική επιστημονική υπολογιστική βασίζεται σε 32-bit ή 64-bit κινητή υποδιαστολή, τα νευρωνικά δίκτυα είναι εκπληκτικά ανθεκτικά σε μικρότερη δεκαδική ακρίβεια (*mantissa*), αρκεί να διατηρείται επαρκές **δυναμικό εύρος (exponent)** ώστε οι μικρές κλίσεις να μην υποχειλίζουν στο μηδέν. Η **Αυτόματη Μικτή Ακρίβεια (Automatic Mixed Precision, AMP)** συνδυάζει το καλύτερο και των δύο κόσμων: διατηρεί ακριβή κύρια βάρη σε **FP32 (Master Weights)** για την αθροιστική ενημέρωση του βελτιστοποιητή, αλλά μετατρέπει αυτόματα τις εμπρόσθιες και ανάστροφες πράξεις πινάκων σε **FP16 ή BF16**, υποδιπλασιάζοντας την κίνηση μνήμης VRAM και επιταχύνοντας **2× έως 4×** τους Tensor Cores!

Επειδή το πρότυπο **FP16** διαθέτει μόλις 5 bits εκθέτη (μέγιστη τιμή 65.504 και κίνδυνο υποχείλισης μικρών κλίσεων κάτω από 6×10^{-8}), το AMP εφαρμόζει **Δυναμική Κλιμάκωση Απώλειας (Dynamic Loss Scaling, `GradScaler`)**: πολλαπλασιάζει την απώλεια με μεγάλο συντελεστή S πριν την οπισθοδιάδοση ώστε οι κλίσεις να μετατοπιστούν στην ασφαλή ζώνη του FP16, και στη συνέχεια τις διαιρεί με S (με έλεγχο `Inf/NaN`) πριν το βήμα ενημέρωσης των FP32 Master Weights. Αντίθετα, το **BF16 (Brain Float 16)** διατηρεί αυτούσια τα **8 bits εκθέτη του FP32** (ίδιο δυναμικό εύρος έως 3.4×10^{38}), καθιστώντας το χρυσό πρότυπο εκπαίδευσης LLMs *χωρίς καμία ανάγκη Loss Scaling*!

Πρότυπο	Δομή Bits (S / E / M)	Μέγιστη Τιμή (Εύρος)	Ταχύτητα Tensor Core	Ρόλος στη Βαθιά Μάθηση
FP32	32-bit (S:1, E:8, M:23)	3.4×10^{38}	1.0× (αναφορά)	Master weights, καταστάσεις AdamW (m_t, v_t), συσσώρευση απώλειας
TF32	19-bit (S:1, E:8, M:10)	3.4×10^{38}	~2.0×	Προεπιλεγμένο πρότυπο GEMM σε Tensor Cores (Ampere/Ada) χωρίς αλλαγή κώδικα
BF16	16-bit (S:1, E:8, M:7)	3.4×10^{38}	2× έως 4×	Χρυσό πρότυπο LLMs και Transformers (ίδιο εύρος με FP32, μηδενική ανάγκη Loss Scaling)
FP16	16-bit (S:1, E:5, M:10)	65.504	2× έως 4×	Υψηλότερη δεκαδική ακρίβεια από BF16, απαιτεί *Dynamic Loss Scaling* (GradScaler)
FP8 (E4M3)	8-bit (S:1, E:4, M:3)	448.0	4× έως 6×	Υπερταχεία εκπαίδευση & εξαγωγή συμπερασμάτων σε GPU Hopper / Blackwell



Σχήμα 8.4

Αγωγός Αυτόματης Μικτής Ακρίβειας (AMP) με FP32 Master Weights, Tensor Core GEMM και Dynamic Loss Scaling, καθώς και σύγκριση δομής bits των προτύπων FP32, TF32, BF16, FP16 και FP8.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Υπολογισμός Ταχύτητας (CPU vs RTX 3060) & Μνήμης VRAM για Μοντέλο 10M Παραμέτρων

Α. Σύγκριση 1.000.000 πράξεων πολλαπλασιασμού-πρόσθεσης (MACs):

- **20-πύρηνη CPU (2,5 GHz, 1 βαθμωτή πράξη/κύκλο/πυρήνα):** *1.000.000 / 20 = 50.000 κύκλοι* → περίπου **20,0 μικρο-δευτερόλεπτα (μs)**.
- **NVIDIA RTX 3060 (3.584 CUDA Cores στα 1,78 GHz):** [*1.000.000 / 3.584*] = *280 κύκλοι* → περίπου **0,16 μs** (πάνω από **125×** ταχύτερη ακατέργαστη παραλληλοποίηση, και ακόμη περισσότερο στους Tensor Cores σε FP16!).

Β. Πόση VRAM καταλαμβάνει ένα μοντέλο 10.000.000 (10M) παραμέτρων;

- **Καθαρό FP32 (4 bytes/αριθμό):** Βάρη = *10M × 4 B* = ***40 MB***, κλίσεις = ***40 MB***, καταστάσεις AdamW (m_t, v_t) = ***80 MB*** → συνολική στατική κατάσταση εκπαίδευσης = **160 MB** (ο περίφημος *κανόνας των 16 bytes ανά παράμετρο*, συν τις ενεργοποιήσεις κάθε batch!).
- **Μικτή Ακρίβεια FP16 / AMP (2 bytes/αριθμό):** Τα βάρη εξαγωγής συμπερασμάτων (inference) πέφτουν στο μισό (**20 MB** αντί 40 MB) και οι ενδιάμεσες ενεργοποιήσεις του batch συρρικνώνονται κατά ~**50%**, επιτρέποντας διπλάσιο `batch_size`! Με τον ίδιο κανόνα, ένα LLM 7 δισεκατομμυρίων παραμέτρων θέλει *7B × 2 B = 14 GB* σε FP16 (γι' αυτό η 4-bit κβαντοποίηση του Κεφαλαίου 10 το χωρά σε ~4 GB VRAM!).

8.3 Περιβάλλον Εργασίας Python 3.12+ & Εργαστήρια Επιτάχυνσης 11 και 12

Για επαναλήψιμη, σταθερή ανάπτυξη σε **Python 3.12+** τηρούμε τέσσερις πρακτικούς κανόνες:

1. **Απομονωμένο εικονικό περιβάλλον:** Δημιουργία `python -m venv .venv` (ή `uv venv`) και εγκατάσταση των εξαρτήσεων από το `requirements.txt`.
2. **Συμβατότητα οδηγών NVIDIA & CUDA:** Στα Windows το PyTorch 2 υποστηρίζει εγγενώς CUDA GPU, ενώ για επιτάχυνση GPU στο TensorFlow συνιστάται Linux ή WSL2.
3. **Δυναμική δέσμευση VRAM:** Ενεργοποίηση `set_memory_growth(gpu, True)` ώστε το TensorFlow να δεσμεύει σταδιακά τη μνήμη της κάρτας γραφικών.
4. **Συγχρονισμένη χρονομέτρηση CUDA & AMP:** Επειδή οι κλήσεις πυρήνων CUDA είναι ασύγχρονες ως προς την CPU, κάθε αξιόπιστη μέτρηση χρόνου απαιτεί `torch.cuda.synchronize()`.

ΕΡΓΑΣΤΗΡΙΟ 11

Συγκριτική Μέτρηση Απόδοσης CPU έναντι CUDA GPU σε PyTorch 2 με AMP & Synchronize

`code/11_gpu_vs_cpu_pytorch.py`

Εκπαιδεύει το ίδιο συνελικτικό δίκτυο δύο σταδίων (`MnistCNN`: 1→32→64 κανάλια, Dropout 25%) στο MNIST αρχικά στην **CPU** και έπειτα στην **CUDA GPU** με `torch.amp.autocast('cuda')`, `GradScaler` και ρητό `torch.cuda.synchronize()`, υπολογίζοντας τον ακριβή συντελεστή επιτάχυνσης.

Προτεινόμενος Πειραματισμός: Μεταβάλετε το `batch_size` από `32` σε `256` και συγκρίνετε την επιτάχυνση: μεγαλύτερα batches αυξάνουν την *αριθμητική ένταση I* (Roofline), τροφοδοτώντας αποδοτικότερα τα warps της GPU.

PYTHON 3.12+ `code/11_gpu_vs_cpu_pytorch.py` Πυρήνας μέτρησης απόδοσης CPU vs CUDA GPU με PyTorch 2 AMP

```python
import time
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import datasets, transforms


class MnistCNN(nn.Module):
    """Συνελικτικό νευρωνικό δίκτυο δύο σταδίων για ταξινόμηση εικόνων 28x28 σε 10 κλάσεις."""

    def __init__(self) -> None:
        """Ορίζει τη βαθμίδα εξαγωγής συνελικτικών χαρακτηριστικών και την κεφαλή ταξινόμησης."""
        super().__init__()
        # Δύο μπλοκ Conv2d + ReLU + MaxPool2d μειώνουν τη χωρική διάσταση από 28x28 σε 7x7
        self.features = nn.Sequential(
            nn.Conv2d(1, 32, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2),
        )
        # Πλήρως συνδεδεμένος ταξινομητής με Dropout 25% για περιορισμό υπερεκπαίδευσης
        self.classifier = nn.Sequential(
            nn.Flatten(),
            nn.Linear(64 * 7 * 7, 128),
            nn.ReLU(inplace=True),
            nn.Dropout(p=0.25),
            nn.Linear(128, 10),
        )

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        """Εκτελεί την εμπρόσθια διέλευση και επιστρέφει τα ακανονικοποίητα logits 10 κλάσεων."""
        return self.classifier(self.features(x))


def benchmark_device(
    device: torch.device,
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/11_gpu_vs_cpu_pytorch.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 12

Δυναμική Διαχείριση Μνήμης VRAM & Μέτρηση Απόδοσης CPU/GPU σε Keras 3 / TensorFlow

`code/12_gpu_vs_cpu_keras.py`

Επιδεικνύει την ορθή διαχείριση πόρων GPU στο TensorFlow/Keras 3 μέσω `tf.config.experimental.set_memory_growth(gpu, True)` και την ρητή τοποθέτηση εκτέλεσης σε συσκευές (`with tf.device('/CPU:0')` έναντι `/GPU:0`), καταγράφοντας τον χρόνο εκπαίδευσης και την καμπύλη ακρίβειας MNIST.

Προτεινόμενος Πειραματισμός: Παρατηρήστε μέσω `nvidia-smi` σε παράλληλο τερματικό τη διαφορά δέσμευσης VRAM με και χωρίς `set_memory_growth`, καθώς και την ανίχνευση συσκευής σε Windows έναντι WSL2/Linux.

PYTHON 3.12+ `code/12_gpu_vs_cpu_keras.py` — Διαμόρφωση memory_growth & benchmark CPU vs GPU σε Keras 3

```python
import time
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import keras
from keras import layers


def configure_gpu_memory_growth() -> list:
    """Ενεργοποιεί τη σταδιακή δέσμευση μνήμης VRAM στις διαθέσιμες φυσικές GPU."""
    gpus = tf.config.list_physical_devices("GPU")
    for gpu in gpus:
        # Αποτρέπει την άμεση δέσμευση όλης της VRAM κατά την εκκίνηση του TensorFlow
        tf.config.experimental.set_memory_growth(gpu, True)
    return gpus


def build_mnist_model() -> keras.Model:
    """Κατασκευάζει και μεταγλωττίζει ελαφρύ συνελικτικό δίκτυο CNN για ταξινόμηση MNIST."""
    model = keras.Sequential(
        [
            keras.Input(shape=(28, 28, 1)),
            layers.Conv2D(32, kernel_size=3, activation="relu"),
            layers.MaxPooling2D(pool_size=2),
            layers.Flatten(),
            layers.Dense(128, activation="relu"),
            layers.Dropout(0.2),
            layers.Dense(10, activation="softmax"),
        ]
    )
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=1e-3),
        loss="sparse_categorical_crossentropy",
        metrics=["accuracy"],
    )
    return model


def run_on_device(
    device_name: str,
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/12_gpu_vs_cpu_keras.py του code.zip)
```

ΚΕΦΑΛΑΙΟ 9

Εσωτερική Αρχιτεκτονική Μεγάλων Γλωσσικών Μοντέλων (LLMs)

Από τη διαμέριση υπολέξεων BPE, τις περιστροφικές ενσωματώσεις RoPE και την κρυφή μνήμη KV-Cache έως τη δειγματοληψία Temperature, Top-P, Min-P και τη συλλογιστική Test-Time Compute.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Διαμέριση Υπολέξεων BPE:** Κατανόηση του άπληστου αλγορίθμου συγχώνευσης ζευγών συμβόλων, της πολυγλωσσικής αποδοτικότητας στα Ελληνικά και της αποφυγής άγνωστων λέξεων.
- **Γεωμετρία Θέσης RoPE:** Μαθηματική ανάλυση της δισδιάστατης μιγαδικής περιστροφής και της αναλλοίωτης σχετικής απόστασης ($m - n$) σε τεράστια παράθυρα συμφραζομένων.
- **Μηχανική Μνήμης KV-Cache & GQA:** Ακριβής υπολογισμός αποτυπώματος VRAM, σύγκριση MHA, GQA και MQA, και διαχωρισμός φάσεων Prefill και Decode.
- **Στρατηγικές Αποκωδικοποίησης & Συλλογισμού:** Εφαρμογή Greedy, Beam Search, Temperature, Top-K, Top-P (Nucleus), Min-P και Test-Time Compute (Εργαστήρια 23 & 24).

9.1 Αυτοπαλίνδρομη Παραγωγή, Φάσεις Prefill/Decode & Αλγόριθμος BPE

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ **Πώς Σκέφτεται ένα LLM: Υπολέξεις, Prefill & Decode με Απλά Λόγια**

Αν ξεκινάτε τώρα το ταξίδι σας στα **Μεγάλα Γλωσσικά Μοντέλα (LLMs)**, κρατήστε μία απλή, θεμελιώδη εικόνα: στην καρδιά του, ένα LLM λειτουργεί σαν έναν εξαιρετικά διαβασμένο συγγραφέα που **προβλέπει ένα κομμάτι λέξης (υπολέξη) τη φορά**, διαλέγοντας κάθε φορά τη συνέχεια που ταιριάζει καλύτερα σε όσα έχουν ήδη γραφτεί. Η λειτουργία του χωρίζεται σε δύο απλές φάσεις που θυμίζουν εξετάσεις στο πανεπιστήμιο: η φάση **Prefill** είναι σαν να διαβάζετε *ολόκληρη την εκφώνηση του θέματος μία φορά με μία ματιά*, ενώ η φάση **Decode** είναι σαν να γράφετε την απάντηση *λέξη προς λέξη*, κρατώντας οργανωμένες πρόχειρες σημειώσεις (**KV-Cache**) ώστε να μη χρειάζεται να ξαναδιαβάζετε από την αρχή ό,τι έχετε ήδη γράψει!

- **Token (Διακριτικό / Υπολέξη):** Το στοιχειώδες «τουβλάκι» κειμένου που βλέπει το μοντέλο (ολόκληρη συχνή λέξη ή συλλαβή, που μετατρέπεται σε έναν ακέραιο αριθμό από τον διαχωριστή **BPE**).
- **RoPE (Περιστροφική Κωδικοποίηση Θέσης):** Ο ψηφιακός «δείκτης σειράς» που επιτρέπει στο δίκτυο να ξεχωρίζει το *«ο σκύλος κυνηγά τη γάτα»* από το *«η γάτα κυνηγά τον σκύλο»*.
- **VRAM & Logits:** Η **VRAM** είναι η υπερταχεία μνήμη επάνω στην κάρτα γραφικών (GPU), ενώ **logits** ονομάζονται οι ακατέργαστες βαθμολογίες προτίμησης που δίνει το μοντέλο σε κάθε υποψήφια επόμενη λέξη πριν μετατραπούν σε ποσοστά πιθανότητας.

Μετά τη θεμελίωση της αρχιτεκτονικής **Transformers** (Κεφάλαιο 5) και της παράλληλης υπολογιστικής ισχύος των **GPU & CUDA** (Κεφάλαιο 8), εξετάζουμε πώς αυτοί οι δύο πυλώνες συνθέτουν τα σύγχρονα **Μεγάλα Γλωσσικά Μοντέλα (Large Language Models, LLMs)**. Κάθε σύγχρονο μοντέλο αρχιτεκτονικής *decoder-only* (όπως τα Gemma 3, Llama 3/4, Qwen 2.5/3, DeepSeek και Gemini) μοντελοποιεί την από κοινού πιθανότητα μιας ακολουθίας διακριτικών (*tokens*) μέσω του κανόνα γινομένου: $P(x_1, \ldots, x_T) = \Pi_{t=1..T} P(x_t \mid x_{<t})$. Κατά την εκτέλεση διακρίνουμε δύο εντελώς διαφορετικές υπολογιστικές φάσεις:

- **Φάση Προπλήρωσης (Prefill):** Ολόκληρη η προτροπή εισόδου (*prompt*) επεξεργάζεται *παράλληλα* σε ένα ενιαίο πέρασμα πολλαπλασιασμού μεγάλων πινάκων (GEMM), αξιοποιώντας στο έπακρο τους Tensor Cores της GPU (φάση περιορισμένη από την υπολογιστική ισχύ, *compute-bound*) και γεμίζοντας την αρχική μνήμη KV-Cache.
- **Φάση Αποκωδικοποίησης (Decode):** Το μοντέλο παράγει *ένα διακριτικό ανά βήμα*, διαβάζοντας κάθε φορά όλους τους πίνακες βαρών και το συσσωρευμένο KV-Cache από τη VRAM. Η φάση αυτή περιορίζεται από το εύρος ζώνης μνήμης (*memory-bandwidth-bound*) και καθορίζει άμεσα την ταχύτητα παραγωγής σε tokens/sec.

Πριν εισέλθει οποιοδήποτε κείμενο στον Transformer, ο **Υπολεξικός Διαχωριστής BPE (Byte-Pair Encoding)** το μετατρέπει σε ακέραιους δείκτες λεξιλογίου. Για έναν προπτυχιακό φοιτητή η διαίσθηση είναι απλή: οι μεμονωμένοι χαρακτήρες παράγουν υπερβολικά μεγάλες ακολουθίες, ενώ ένα στατικό λεξικό ολόκληρων λέξεων αποτυγχάνει σε σπάνιες κλίσεις, ονόματα ή κώδικα. Ο άπληστος αλγόριθμος εκπαίδευσης BPE γεφυρώνει ιδανικά τα δύο άκρα σε τέσσερα βήματα:

1. **Αρχικοποίηση Επιπέδου Byte / Χαρακτήρα:** Κάθε λέξη διασπάται στα στοιχειώδη σύμβολά της (ή στα 256 βασικά UTF-8 bytes) με ειδικό δείκτη ορίου λέξης (`</w>`), εξαλείφοντας οριστικά το σφάλμα αγνώστων όρων (`<UNK>`).
2. **Καταμέτρηση Διγραμμάτων:** Μετρώνται οι συχνότητες όλων των γειτονικών ζευγών συμβόλων `(a, b)` σε ολόκληρο το σώμα εκπαίδευσης.
3. **Άπληστη Συγχώνευση (Greedy Merge):** Το συχνότερο ζεύγος `(a*, b*)` συγχωνεύεται σε ένα νέο σύμβολο `a*b*`, ο κανόνας καταγράφεται σε διατεταγμένη λίστα και αντικαθίσταται παντού στο λεξιλόγιο.
4. **Επανάληψη έως το Στόχο |V|:** Η διαδικασία επαναλαμβάνεται μέχρι το λεξιλόγιο να φτάσει τυπικά τις 100.000 έως 256.000 υπολέξεις (π.χ. 128K στο Llama 3 και 256K στο Gemma 3). Τα σύγχρονα ευρέα πολυγλωσσικά λεξιλόγια μειώνουν δραστικά τη *γονιμότητα διακριτικών (token fertility)* στα Ελληνικά, επιτρέποντας την κωδικοποίηση ελληνικών επιστημονικών κειμένων με 2× έως 3× λιγότερα tokens σε σύγκριση με παλαιότερους αγγλοκεντρικούς διαχωριστές.

9.2 Περιστροφικές Ενσωματώσεις Θέσης (RoPE): Μιγαδική Γεωμετρία Μακρού Πλαισίου

Επειδή ο μηχανισμός αυτοπροσοχής δεν ξεχωρίζει από μόνος του τη σειρά των λέξεων (είναι αναλλοίωτος σε μεταθέσεις, *permutation equivariant*), χρειάζεται ακριβή κωδικοποίηση της θέσης τους. Οι κλασικές προσθετικές ενσωματώσεις θέσης φθείρονται όταν η ακολουθία ξεπεράσει το μήκος εκπαίδευσης. Αντίθετα, η μέθοδος **RoPE (Rotary Positional Embedding, Su et al.)** κωδικοποιεί τη θέση m περιστρέφοντας κάθε δισδιάστατο ζεύγος συνιστωσών (x_{2i}, x_{2i+1}) των διανυσμάτων Query (q) και Key (k) στο μιγαδικό επίπεδο κατά γωνία $m \cdot \theta_i$, όπως οι δείκτες ενός ρολογιού πολλαπλών συχνοτήτων:

$$\mathbf{R_{\Theta,m}^{(i)}\,[x_{2i},\,x_{2i+1}]^T = [x_{2i}\cos(m\theta_i) - x_{2i+1}\sin(m\theta_i),\; x_{2i}\sin(m\theta_i) + x_{2i+1}\cos(m\theta_i)]^T \;\mid\; \theta_i = b^{-2i/d_{head}}}$$

$$\mathbf{\langle R_m\,q,\, R_n\,k\rangle = q^T\,R_m^T\,R_n\,k = q^T\,R_{n-m}\,k}$$

Επειδή οι πίνακες περιστροφής είναι ορθογώνιοι ($R_m^T R_n = R_{n-m}$), το εσωτερικό γινόμενο προσοχής εξαρτάται **αποκλειστικά από τη σχετική απόσταση (m − n)**. Κλιμακώνοντας τη βάση συχνότητας b (π.χ. από 10.000 σε 500.000 με μεθόδους NTK / YaRN), τα σύγχρονα LLMs επεκτείνονται ομαλά σε παράθυρα 128K έως 1M tokens.

9.3 Αιτιώδης Μάσκα, Κρυφή Μνήμη KV-Cache & Σύγκριση MHA, GQA, MQA

Κατά την αυτοπαλίνδρομη παραγωγή, η κατωτριγωνική αιτιώδης μάσκα ($-\infty$ άνω της κυρίας διαγωνίου πριν τη Softmax) απαγορεύει σε κάθε token να αντλεί πληροφορία από το μέλλον. Εάν σε κάθε νέο token t επαναϋπολογίζαμε τους τανυστές K και V για όλα τα προηγούμενα $t-1$ διακριτικά, το κόστος παραγωγής μιας απάντησης μήκους S θα κλιμακωνόταν κυβικά. Αποθηκεύοντας τα ήδη υπολογισμένα ζεύγη (K, V) στη μνήμη VRAM της GPU (**KV-Cache**), κάθε βήμα υπολογίζει μόνο το νέο διάνυσμα q_t, k_t, v_t και η πολυπλοκότητα ανά token πέφτει από `O(S²)` σε γραμμική `O(S)`.

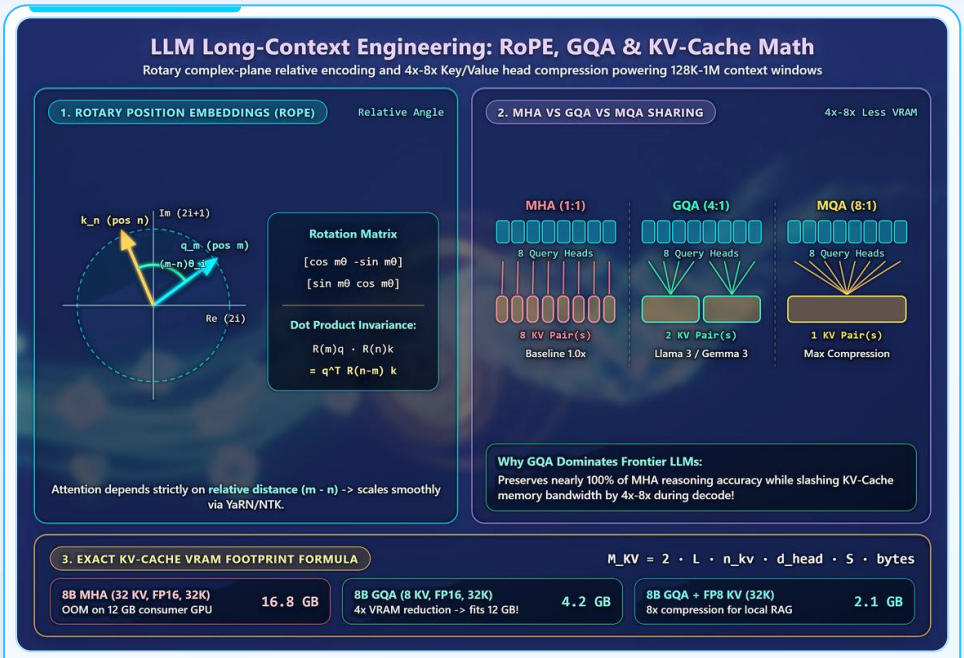
Ωστόσο, σε μεγάλα παράθυρα συμφραζομένων η ίδια η μνήμη KV-Cache γίνεται ο κύριος περιορισμός VRAM. Το ακριβές μέγεθός της σε bytes δίνεται από την εξίσωση:

$$\mathbf{M_{KV} = 2 \cdot L \cdot n_{kv} \cdot d_{head} \cdot S \cdot B \cdot bytes}$$

Όπου 2 = τανυστές Key και Value, L = αριθμός επιπέδων Transformer, n_{kv} = κεφαλές Key/Value ανά επίπεδο, d_{head} = διάσταση ανά κεφαλή, S = μήκος ακολουθίας (context length), B = μέγεθος δέσμης (batch size) και bytes = ακρίβεια αποθήκευσης (2 για FP16/BF16, 1 για FP8/INT8).

Αριθμητικό Παράδειγμα (Μοντέλο 8B: $L = 32$, $d_{head} = 128$, $B = 1$, ακρίβεια FP16 = 2 bytes): Με κλασική **Multi-Head Attention (MHA)** και $n_{kv} = 32$ κεφαλές, σε πλαίσιο **8K** ($S = 8.192$) το KV-Cache απαιτεί $2 \times 32 \times 32 \times 128 \times 8.192 \times 2 = 4,29$ *GB* (4,00 GiB), στα **32K** εκτινάσσεται στα *17,18 GB* (16,00 GiB, προκαλώντας σφάλμα Out-of-Memory σε GPU 12 GB!), ενώ στα **128K** ($S = 131.072$) φτάνει τα δυσβάσταχτα **68,72 GB (64,00 GiB)** μόνο για το cache! Αντίθετα, υιοθετώντας **Grouped-Query Attention (GQA 4:1)** με $n_{kv} = 8$ κοινόχρηστες κεφαλές KV (όπως στο Llama 3 8B), η μνήμη μειώνεται κατά **4×**: μόλις **1,00 GiB (1,07 GB)** στα 8K, **4,00 GiB (4,29 GB)** στα 32K και **16,00 GiB (17,18 GB)** στα 128K (ή μόλις **8,00 GiB** με κβαντοποίηση KV-Cache σε FP8), χωρώντας άνετα σε τοπικές πανεπιστημιακές GPU.

Αρχιτεκτονική Προσοχής	Αναλογία Κεφαλών ($n_q : n_{kv}$)	Μέγεθος KV-Cache (Σχετικό)	Ποιότητα Συλλογισμού & Εκπαίδευσης	Ενδεικτικά Μοντέλα
MHA (Multi-Head Attention)	1 : 1 (π.χ. 32 Q → 32 KV)	1.00× (Βάση αναφοράς, μέγιστο VRAM)	Μέγιστη εκφραστικότητα αλλά βαρύ εύρος ζώνης	GPT-3, Llama 1/2 (7B/13B)
GQA (Grouped-Query Attention)	4 : 1 έως 8 : 1 (π.χ. 32 Q → 8 KV)	0.25× έως 0.125× (4× έως 8× συμπίεση)	Διατηρεί ~99.8% της ποιότητας MHA με υψηλή ταχύτητα	Llama 3, Gemma 3, Qwen 2.5/3, Mistral
MQA (Multi-Query Attention)	n_q : 1 (όλες οι Q → 1 KV)	1 / n_q (Μέγιστη δυνατή συμπίεση)	Πιθανή ήπια απώλεια ευστάθειας σε σύνθετες εργασίες	PaLM, Falcon, StarCoder
MLA (Multi-Head Latent Attention)	Χαμηλής τάξης λανθάνουσα συμπίεση c_{KV}	<0.10× (Υπέρ-συμπαγές λανθάνον cache)	Ισχυρή αναπαράσταση με ελάχιστο αποτύπωμα μνήμης	DeepSeek-V2 / V3 / R1



Σχήμα 9.1

Μηχανική μακρού πλαισίου LLMs: περιστροφικές ενσωματώσεις RoPE στο μιγαδικό επίπεδο με αναλλοίωτη σχετική γωνία (m - n)θ_i, σύγκριση διαμοιρασμού κεφαλών MHA/GQA/MQA και ακριβής κλιμάκωση μνήμης KV-Cache VRAM.

9.4 Μαθηματικά Αποκωδικοποίησης (Decoding), Δειγματοληψία & Test-Time Compute

Στο τελικό στρώμα εξόδου το μοντέλο παράγει ένα διάνυσμα ακατέργαστων βαθμολογιών (*logits*) z_i για κάθε διακριτικό του λεξιλογίου. Η **Θερμοκρασία (Temperature, T)** κλιμακώνει τα logits πριν τη συνάρτηση Softmax, ελέγχοντας την εντροπία Shannon της κατανομής και την ισορροπία μεταξύ ντετερμινιστικής αυστηρότητας και δημιουργικής

ποικιλίας:

$$P(w_i) = \exp(z_i / T) / \Sigma_j \exp(z_j / T) \quad | \quad H(P) = -\Sigma_i P(w_i) \log_2 P(w_i)$$

Χαμηλή θερμοκρασία (T = 0.2) ελαχιστοποιεί την εντροπία H(P) για αξιόπιστο κώδικα και αποδείξεις, ενώ υψηλότερη (T = 0.8 έως 1.1) διευρύνει το φάσμα δημιουργικών επιλογών.

Στη μηχανική παραγωγής LLMs επιλέγουμε τη στρατηγική αποκωδικοποίησης ανάλογα με τη φύση του προβλήματος:

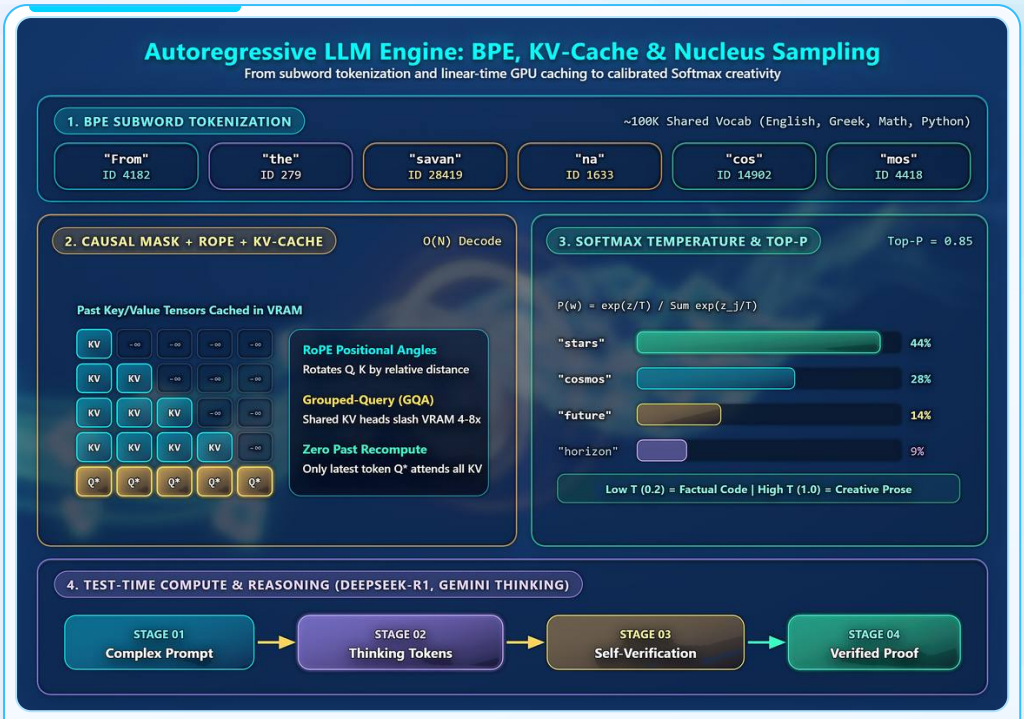
- **Άπληστη Αποκωδικοποίηση (Greedy Decoding, *T → 0*):** Επιλέγει πάντοτε το `argmax(z_i)`. Είναι πλήρως επαναλήψιμη, αλλά σε ελεύθερο κείμενο συχνά παγιδεύεται σε επαναλαμβανόμενους βρόχους φράσεων.
- **Αναζήτηση Δέσμης (Beam Search, πλάτος *B*):** Διατηρεί παράλληλα τις *B* ακολουθίες με τη μέγιστη αθροιστική λογαριθμική πιθανότητα. Υπερέχει σε μηχανική μετάφραση και αναγνώριση ομιλίας, αλλά τείνει σε γενικόλογες απαντήσεις στον ανοιχτό διάλογο.
- **Περικοπή Top-K:** Κρατά αυστηρά τα *K* πιθανότερα tokens (π.χ. *K = 40*) και επανακανονικοποιεί. Εάν η κατανομή είναι πολύ οξεία, το σταθερό *K* επιτρέπει απίθανες λέξεις, ενώ αν είναι επίπεδη, αποκόπτει έγκυρες εναλλακτικές.
- **Πυρηνική Δειγματοληψία Top-P (Nucleus Sampling):** Επιλέγει δυναμικά το ελάχιστο υποσύνολο υποψηφίων των οποίων το άθροισμα πιθανοτήτων φτάνει το κατώφλι *P* (π.χ. *P = 0.85*), προσαρμόζοντας αυτόματα το μέγεθος του πυρήνα στην τοπική εντροπία του συμφραζομένου.
- **Κλιμακούμενη Αποκοπή Min-P:** Διατηρεί μόνο τα διακριτικά με $P(w_i) \geq p_{min} \cdot P_{max}$ (π.χ. $p_{min} = 0.05$). Κλιμακώνοντας το κατώφλι αναλογικά προς τη βεβαιότητα της κορυφαίας λέξης P_{max}, διατηρεί υψηλή συνοχή ακόμη και σε αυξημένες θερμοκρασίες.
- **Συλλογιστική Χρόνου Εκτέλεσης (Test-Time Compute & Chain-of-Thought):** Τα μετωπικά μοντέλα συλλογισμού (DeepSeek-R1, Gemini Thinking) μετατοπίζουν μέρος της κλιμάκωσης από την εκπαίδευση στον χρόνο απόκρισης, παράγοντας εσωτερικά διακριτικά σκέψης (*thinking tokens*), αυτοέλεγχο λαθών και αναζήτηση πολλαπλών μονοπατιών πριν από την τελική επιστημονική απάντηση.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Υπολογισμός Softmax με Θερμοκρασία (T = 0.5 vs T = 1.5) & Περικοπή Top-P (P = 0.90)

Έστω ότι το μοντέλο συμπληρώνει τη φράση *«Κοιτώντας ψηλά βλέπω τα ...»* και παράγει ακατέργαστα logits για τρεις υποψήφιες λέξεις: `"άστρα": 4.0`, `"νύχτα": 3.0` και `"θάλασσα": 1.0`. Για αριθμητική ευστάθεια αφαιρούμε πάντα το μέγιστο κλιμακωμένο logit πριν από την εκθετική e^x (οι λόγοι πιθανοτήτων μένουν απολύτως ίδιοι):

1. **Χαμηλή Θερμοκρασία `T = 0.5` (Οξεία / Αυστηρή):** Διαιρούμε με 0.5 → `[8.0, 6.0, 2.0]`, αφαιρούμε το 8.0 → `[0.0, -2.0, -6.0]`. Οι εκθετικές είναι $e^0 = 1.000$, $e^{-2} = 0.135$, $e^{-6} = 0.0025$ (άθροισμα **1.1378**). Διαιρώντας με το άθροισμα προκύπτουν: **«άστρα» = 87.9%** (~88%), **«νύχτα» = 11.9%**, **«θάλασσα» = 0.2%**. Το μοντέλο κλειδώνει σχεδόν αποκλειστικά στην κορυφαία λέξη.
2. **Υψηλή Θερμοκρασία `T = 1.5` (Δημιουργική):** Διαιρούμε με 1.5 → `[2.67, 2.00, 0.67]`, αφαιρούμε το 2.67 → `[0.00, -0.67, -2.00]`. Οι εκθετικές είναι *1.000*, *0.513*, *0.135* (άθροισμα **1.6488**), δίνοντας πολύ πιο ανοιχτή κατανομή: **«άστρα» = 60.7%**, **«νύχτα» = 31.1%** και **«θάλασσα» = 8.2%**.
3. **Περικοπή Top-P (`P = 0.90`) στο `T = 1.5`:** Αθροίζουμε τις ταξινομημένες πιθανότητες από την κορυφή: *60.7% < 90%* (κρατάμε «άστρα»), προσθέτουμε «νύχτα» → *60.7% + 31.1% = 91.8% ≥ 90%* και **σταματάμε**! Η λέξη «θάλασσα» (8.2%) απορρίπτεται, και οι δύο επιζήσασες επανακανονικοποιούνται σε **66.1% («άστρα»)** και **33.9% («νύχτα»)**, συνδυάζοντας δημιουργικότητα χωρίς άσχετες αστοχίες.



Σχήμα 9.2

Ολοκληρωμένος αγωγός παραγωγής LLM: διαμέριση υπολέξεων BPE, αιτιώδης μάσκα με RoPE & KV-Cache στη VRAM, δειγματοληψία Softmax (Temperature, Top-K & Top-P) και συλλογιστική Test-Time Compute.

9.5 Εργαστήρια 23 & 24: BPE, Δειγματοληψία & Mini-GPT Decoder από το Μηδέν

ΕΡΓΑΣΤΗΡΙΟ 23 **Υπολέξεις BPE, Δειγματοληψία Temperature / Top-K / Top-P & Εντροπία Shannon**

`code/23_bpe_tokenizer_and_sampling.py`

Υλοποιεί από το μηδέν σε καθαρή Python & NumPy τον άπληστο αλγόριθμο εκπαίδευσης και κωδικοποίησης **Byte-Pair Encoding (BPE)** με 18 διαδοχικές συγχωνεύσεις, καθώς και πλήρη δειγματολήπτη Softmax με κλιμάκωση Θερμοκρασίας (*T*), περικοπή **Top-K**, φιλτράρισμα **Top-P (Nucleus)** και ακριβή υπολογισμό της εντροπίας Shannon *H(P)* σε bits.

Προτεινόμενος Πειραματισμός: Αυξήστε το `num_merges` από 18 σε 35 και παρατηρήστε πώς μειώνεται το μήκος των tokens, ή δοκιμάστε `temperature=0.1` έναντι `1.5` καταγράφοντας την απότομη αύξηση της εντροπίας *H* σε bits.

PYTHON 3.12+ `code/23_bpe_tokenizer_and_sampling.py` Εργαστήριο 23: Υπολέξεις BPE, κλιμάκωση Softmax με Temperature/Top-P & εντροπία Shannon

```python
from collections import Counter
import math
import numpy as np


# === 1. Εκπαίδευση & Κωδικοποίηση Υπολεξικού Διαχωριστή BPE ===

def train_bpe(corpus: list[str], num_merges: int = 18):
    """Εκμαθαίνει επαναληπτικά κανόνες συγχώνευσης BPE ενώνοντας το συχνότερο γειτονικό ζεύγος."""
    vocab = Counter()
    for sentence in corpus:
        for word in sentence.lower().split():
            # Προσθήκη ειδικού τερματικού συμβόλου </w> για οριοθέτηση τέλους λέξης
            vocab[" ".join(list(word)) + " </w>"] += 1

    merges = []
    for _ in range(num_merges):
        # Καταμέτρηση συχνότητας όλων των γειτονικών διγραμμάτων συμβόλων στο σώμα κειμένων
        pairs = Counter()
        for word, freq in vocab.items():
```

```
            syms = word.split()
            for i in range(len(syms) - 1):
                pairs[(syms[i], syms[i + 1])] += freq
        if not pairs:
            break
        best = max(pairs, key=pairs.get)
        merges.append(best)
        bigram = " ".join(best)
        fused = "".join(best)
        # Αντικατάσταση του επικρατέστερου ζεύγους σε ολόκληρο το λεξιλόγιο
        vocab = Counter({w.replace(bigram, fused): f for w, f in vocab.items()})
    return merges


def encode_bpe(text: str, merges: list[tuple[str, str]]) -> list[str]:
    """Κατατέμνει αυθαίρετο κείμενο σε υπολεξικά σύμβολα εφαρμόζοντας διαδοχικά τους κανόνες BPE."""
    tokens = []
    for word in text.lower().split():
        syms = list(word) + ["</w>"]
        for a, b in merges:
            i = 0
            while i < len(syms) - 1:
                if syms[i] == a and syms[i + 1] == b:
                    syms[i:i + 2] = [a + b]
                else:
                    i += 1
        tokens.extend(syms)
    return tokens


# === 2. Δειγματοληψία Softmax με Θερμοκρασία, Top-K, Top-P & Εντροπία Shannon ===

def sample_next_token(logits: np.ndarray, temperature: float = 1.0, top_k: int = 0, top_p: float = 1.0):
    """Εφαρμόζει κλιμάκωση θερμοκρασίας, περικοπή Top-K, φιλτράρισμα Top-P και υπολογίζει εντροπία."""
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/23_bpe_tokenizer_and_sampling.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 24

Αυτοπαλίνδρομος Αποκωδικοποιητής Mini-GPT, Αιτιώδης Μάσκα & Υπολογιστής KV-Cache

`code/24_mini_gpt_from_scratch.py`

Κατασκευάζει και εκπαιδεύει ζωντανά έναν πλήρη **Mini-GPT Decoder** (με Pre-LayerNorm, κατωτριγωνική αιτιώδη μάσκα αποκλεισμού μέλλοντος και αυτοπαλίνδρομη παραγωγή χαρακτήρων σε PyTorch, με αυτόματη εναλλακτική καθαρής NumPy), ενώ υπολογίζει την ακριβή κατανάλωση GPU VRAM του **KV-Cache** με GQA ($n_{kv} = 8$) στα 8.192 και 32.768 tokens.

Προτεινόμενος Πειραματισμός: Τροποποιήστε την κλήση `kv_cache_memory_gb()` συγκρίνοντας `n_kv_heads=32` (MHA) έναντι `n_kv_heads=8` (GQA) σε `seq_len=131072` (128K) και μετρήστε τη διαφορά σε GiB VRAM.

PYTHON 3.12+ `code/24_mini_gpt_from_scratch.py` Εργαστήριο 24: Αιτιώδης αυτο-προσοχή Mini-GPT & ακριβής υπολογισμός KV-Cache VRAM

```
import math
import numpy as np


# === 1. Σώμα Κειμένου Εκπαίδευσης & Υπολογιστής Μνήμης KV-Cache ===

CORPUS = (
    "deep learning turns raw data into structured understanding. "
    "transformers connect every token through causal self attention. "
    "open source models bring transparent intelligence to every lab. "
)


def kv_cache_memory_gb(n_layers=32, n_kv_heads=8, head_dim=128, seq_len=8192, bytes_per_elem=2):
    """Υπολογίζει την ακριβή μνήμη GPU VRAM της κρυφής μνήμης KV-Cache (με GQA) σε GB."""
    # Ο παράγοντας 2 αντιστοιχεί στους δύο αποθηκευμένους τανυστές Κλειδιών (K) και Τιμών (V)
```

```
    total_bytes = 2 * n_layers * n_kv_heads * head_dim * seq_len * bytes_per_elem
    return total_bytes / (1024 ** 3)


# === 2. Εναλλακτική Υλοποίηση Αιτιώδους Mini-GPT σε Καθαρή NumPy ===

def run_numpy_mini_gpt():
    """Εκτελεί διαφανή επίδειξη αιτιώδους Self-Attention αποκωδικοποιητή αποκλειστικά σε NumPy."""
    rng = np.random.default_rng(42)
    chars = sorted(set(CORPUS))
    stoi = {c: i for i, c in enumerate(chars)}
    itos = {i: c for c, i in stoi.items()}
    V, D, T = len(chars), 24, 16

    # Τυχαία αρχικοποίηση πινάκων ενσωμάτωσης tokens, θέσης, προβολών Q/K/V και κεφαλής εξόδου
    W_tok = rng.normal(0, 0.2, (V, D))
    W_pos = rng.normal(0, 0.1, (T, D))
    Wq = rng.normal(0, 0.2, (D, D))
    Wk = rng.normal(0, 0.2, (D, D))
    Wv = rng.normal(0, 0.2, (D, D))
    W_head = rng.normal(0, 0.2, (D, V))

    # Εμπειρική προκατάληψη μεταβάσεων διγραμμάτων ώστε η επίδειξη χωρίς εκπαίδευση να παράγει αναγνώσιμες
λέξεις
    trans = np.zeros((V, V), dtype=np.float64)
    for a, b in zip(CORPUS[:-1], CORPUS[1:]):
        trans[stoi[a], stoi[b]] += 2.5

    def forward_causal(idx_seq):
        """Εμπρόσθιο πέρασμα αιτιώδους αυτοπροσοχής με ανωτριγωνική μάσκα αποκλεισμού μέλλοντος."""
        seq = idx_seq[-T:]
        L = len(seq)
        x = W_tok[seq] + W_pos[:L]
        Q, K, Vm = x @ Wq, x @ Wk, x @ Wv
        att = (Q @ K.T) / math.sqrt(D)
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/24_mini_gpt_from_scratch.py του code.zip)
```

ΚΕΦΑΛΑΙΟ 10

Η Επανάσταση των Ανοιχτών Μοντέλων (Open-Source AI) & Κβαντοποίηση

Οικογένειες ανοιχτών βαρών (Gemma 3, Llama, Qwen, DeepSeek, Mistral), θεωρία PTQ/QAT, ομαδική 4-bit κβαντοποίηση NF4/AWQ/GPTQ/GGUF, όριο εύρους ζώνης μνήμης και τοπική εκτέλεση Ollama.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Οικοσύστημα Open-Weight LLMs:** Αξιολόγηση ιδιωτικότητας, επιστημονικής επαναληψιμότητας, ασφαλών μορφών `safetensors` / `GGUF` και μηχανών εκτέλεσης (Ollama, llama.cpp, vLLM).
- **Μαθηματικά Κβαντοποίησης (PTQ vs QAT):** Διαφορές συμμετρικής, ασύμμετρης, ομαδικής ($g = 64/128$) και ποσοστημοριακής κβαντοποίησης **NormalFloat4 (NF4)**.
- **Ακραία Κανάλια (Outliers):** Κατανόηση του φαινομένου των ακραίων ενεργοποιήσεων και σύγκριση των αλγορίθμων **LLM.int8()**, **GPTQ**, **AWQ** και **GGUF Q4_K_M**.
- **Φυσική Ταχύτητας Παραγωγής:** Ανάλυση του ορίου εύρους ζώνης μνήμης (`tok/s ≈ BW / Model_Bytes`) και πρακτική επαλήθευση στα Εργαστήρια 25 & 26.

10.1 Κλειστά APIs Νέφους έναντι Ανοιχτών Μοντέλων (Open-Weight LLMs)

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Γιατί Χωράει ένα LLM στον Προσωπικό μας Υπολογιστή; Η Αναλογία του Χάρακα

Φανταστείτε ότι μετράτε το μήκος χιλιάδων ξύλινων σανίδων και σημειώνετε κάθε μέτρηση με δέκα δεκαδικά ψηφία (π.χ. `12.4839201475 cm`). Εάν στρογγυλοποιήσετε κάθε αριθμό στο πλησιέστερο χιλιοστό (`12.5 cm`), **χωράτε 4 φορές περισσότερες μετρήσεις στην ίδια σελίδα** και τις διαβάζετε 4 φορές ταχύτερα, ενώ για την κατασκευή ενός επίπλου η απώλεια ακρίβειας είναι εντελώς αμελητέα! Ακριβώς αυτή είναι η ιδέα της **Κβαντοποίησης (Quantization)**: συμπιέζει τα δισεκατομμύρια βάρη ενός ανοιχτού μοντέλου από 16-bit σε 4-bit, ώστε ένα σύστημα που κανονικά απαιτούσε ακριβό διακομιστή να τρέχει τοπικά και 100% ιδιωτικά σε μία φοιτητική GPU.

- **Open-Weight Μοντέλο:** Ένα LLM του οποίου τα εκπαιδευμένα βάρη διατίθενται ελεύθερα για λήψη και τοπική εκτέλεση χωρίς σύνδεση στο διαδίκτυο (offline).
- **FP16 έναντι INT4:** Το **FP16** αποθηκεύει κάθε βάρος σε 2 bytes (16 bits), ενώ το **INT4** σε μόλις μισό byte (4 bits, δηλαδή 16 διακριτές στάθμες).
- **GGUF, Ollama & Perplexity:** Το `GGUF` είναι ένα ενιαίο «πακέτο» κβαντοποιημένου μοντέλου, το **Ollama** η τοπική εφαρμογή που το εκτελεί με μία εντολή, και η **Perplexity** το μέτρο «αμηχανίας» της πρόβλεψης (όσο χαμηλότερη, τόσο πιο ακριβές το μοντέλο).

Η οικολογία των σύγχρονων LLMs διακρίνεται σε δύο συμπληρωματικούς κόσμους: τα **ιδιόκτητα μετωπικά συστήματα νέφους** (όπως τα Gemini, Claude και GPT) που προσφέρουν τεράστια παράθυρα συμφραζομένων χωρίς ανάγκη τοπικού υλικού, και τη ραγδαία αναπτυσσόμενη οικογένεια **ανοιχτών βαρών (Open-Weight Models)** όπως τα **Google Gemma 3**, **Meta Llama 3 & 4**, **Alibaba Qwen 2.5 & 3**, **DeepSeek R1** και **Mistral**. Τα βάρη τους διανέμονται ελεύθερα στο Hugging Face Hub σε ασφαλή αρχεία `safetensors` (που αποτρέπουν την εκτέλεση κακόβουλου κώδικα pickle και φορτώνονται άμεσα από τον δίσκο μέσω απευθείας χαρτογράφησης μνήμης, *memory-mapping*) και σε ενιαία κβαντοποιημένα αρχεία `GGUF` για τοπικές μηχανές **Ollama / llama.cpp** ή εξυπηρετητές υψηλής διαμεταγωγής **vLLM / SGLang** με δυναμική σελιδοποίηση KV-Cache (*PagedAttention*).

Κριτήριο Αξιολόγησης	Κλειστά APIs Νέφους	Τοπικά Open-Weight Μοντέλα (Ollama / vLLM)
Ιδιωτικότητα & Κυριαρχία Δεδομένων	Οι προτροπές μεταδίδονται σε εξωτερικούς διακομιστές	100% τοπική εκτέλεση offline, μηδενική διαρροή ευαίσθητων δεδομένων
Κόστος σε Μεγάλη Κλίμακα	Συνεχής χρέωση ανά εκατομμύριο tokens εισόδου/εξόδου	Μηδενικό οριακό κόστος ανά token πάνω σε υπάρχουσες GPUs
Επιστημονική Επαναληψιμότητα	Τα βάρη μπορούν να αλλάξουν αθόρυβα χωρίς ειδοποίηση	Κλειδωμένα, ελέγξιμα βάρη ιδανικά για πανεπιστημιακή έρευνα
Εξειδικευμένη Προσαρμογή	Περιορισμένες επιλογές μέσω εξωτερικού API	Πλήρης έλεγχος βαρών, κβαντοποίησης και τοπικού LoRA/QLoRA

10.2 Θεωρία Κβαντοποίησης: PTQ vs QAT, Ομαδική INT4 & NormalFloat4 (NF4)

Ένα ανοιχτό μοντέλο 8 δισεκατομμυρίων παραμέτρων σε εγγενή ακρίβεια 16-bit (FP16/BF16) απαιτεί περίπου **16.8 GB μνήμης VRAM**, ξεπερνώντας τις δυνατότητες μιας τυπικής φοιτητικής ή εργαστηριακής GPU 12 GB (όπως μια NVIDIA RTX 3060). Η **Κβαντοποίηση (Quantization)** απεικονίζει τα συνεχή βάρη κινητής υποδιαστολής σε ένα μικρό διακριτό σύνολο ακεραίων επιπέδων (π.χ. 256 επίπεδα σε 8-bit ή μόλις 16 επίπεδα σε 4-bit), εκμεταλλευόμενη το γεγονός ότι τα εκπαιδευμένα νευρωνικά βάρη ακολουθούν ομαλές κωδωνοειδείς κατανομές.

Διακρίνουμε δύο βασικές οικογένειες μεθοδολογίας: την **Μετα-Εκπαιδευτική Κβαντοποίηση (PTQ, Post-Training Quantization)**, η οποία συμπιέζει ένα ήδη εκπαιδευμένο μοντέλο σε λίγα λεπτά με ελάχιστα δεδομένα βαθμονόμησης και μηδενικό κόστος επανεκπαίδευσης, και την **Κβαντοποίηση κατά την Εκπαίδευση (QAT, Quantization-Aware Training)**, όπου προσομοιωμένος θόρυβος στρογγυλοποίησης ενσωματώνεται στη διαδικασία οπισθοδιάδοσης μέσω του *Straight-Through Estimator (STE)*. Ενώ η QAT είναι απαραίτητη κάτω από τα 3 bits (όπως τα τριαδικά μοντέλα 1.58-bit), στο καθεστώς των **4 bits** οι σύγχρονες τεχνικές **Ομαδικής Κβαντοποίησης (Group-Wise PTQ, *g = 64* ή *128*)** πετυχαίνουν σχεδόν μηδενική απώλεια ποιότητας:

$$\textbf{Συμμετρική (Group-g): } s_g = \max(|W_g|) / 7 \;\mid\; Q_g = \text{clip}(\text{round}(W_g / s_g), -7, +7) \;\mid\; \hat{W}_g = s_g \cdot Q_g$$

$$\textbf{Ασύμμετρη: } s = (w_{max} - w_{min}) / (2^b - 1), \;\; z = \text{round}(-w_{min} / s) \;\mid\; b_{eff} = b + 16 / g$$

Η ασύμμετρη κβαντοποίηση προσθέτει ένα σημείο μηδενός z για ασύμμετρες κατανομές, ενώ η ομαδική Group-64 INT4 αποθηκεύει μία κλίμακα FP16 ανά 64 βάρη (b_{eff} = 4 + 16/64 = 4.25 bits/βάρος), απομονώνοντας τοπικά τις ακραίες τιμές και διατηρώντας ομοιότητα συνημιτόνου >98.9%.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Κβαντοποίηση 4 Πραγματικών Βαρών FP16 σε Συμμετρικό INT4 & Μέγεθος Μοντέλου 8B

Ας εφαρμόσουμε με το χέρι τη συμμετρική κβαντοποίηση 4-bit (ακέραιοι από **−7 έως +7**) σε τέσσερα βάρη `W = [-0.84, 0.21, 0.63, 1.05]`:

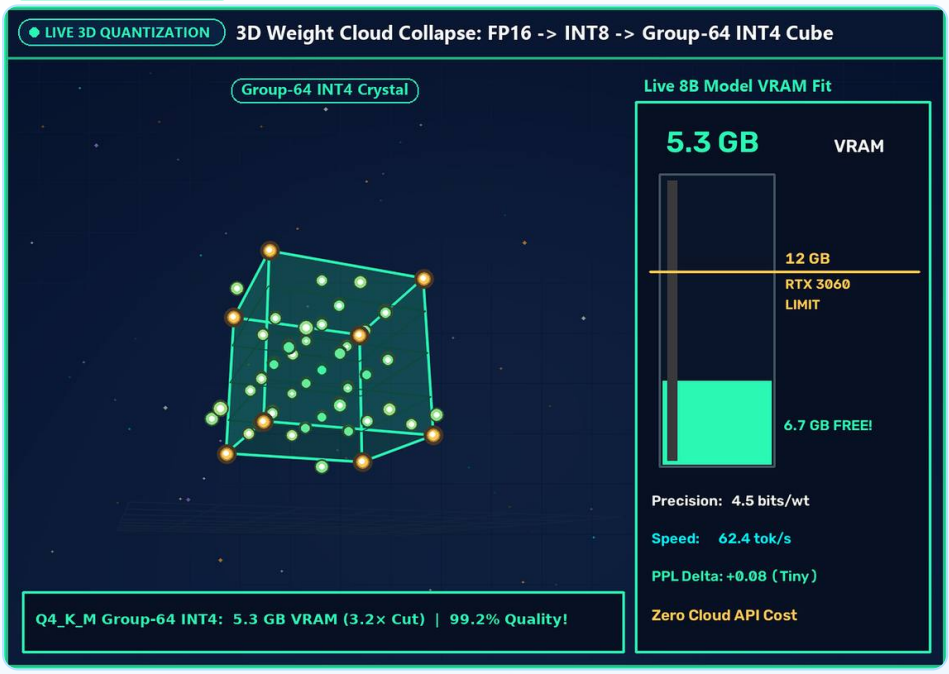
1. **Συντελεστής Κλίμακας (Scale):** Βρίσκουμε τη μέγιστη απόλυτη τιμή *max(|W|) = 1.05* και διαιρούμε με 7 → ***s = 1.05 / 7 = 0.15***.
2. **Συμπίεση σε Ακεραίους 4-bit:** Διαιρούμε κάθε βάρος με *0.15* και στρογγυλοποιούμε: `-0.84 / 0.15 = -5.6 → -6`, `0.21 / 0.15 = 1.4 → +1`, `0.63 / 0.15 = 4.2 → +4`, `1.05 / 0.15 = 7.0 → +7`. Αποθηκεύουμε μόλις τέσσερις 4-bit ακεραίους `q = [-6, +1, +4, +7]` και μία κλίμακα *s = 0.15*!
3. **Αποκβαντοποίηση & Σφάλμα MSE:** Πολλαπλασιάζοντας ξανά με *0.15* παίρνουμε `Ŵ = [-0.90, 0.15, 0.60, 1.05]`. Τα τετράγωνα σφαλμάτων είναι *0.0036 + 0.0036 + 0.0009 + 0.0000 = 0.0081*, άρα το μέσο τετραγωνικό σφάλμα είναι αμελητέο: **MSE = 0.0081 / 4 = 0.0020**!
4. **Αποτύπωμα Μνήμης Μοντέλου 8B (8×10^9 παράμετροι):** Σε **FP16** (2 bytes/βάρος) απαιτεί *8 × 2 = **16 GB*** (~16.8 GB με μεταδεδομένα, αδύνατο να χωρέσει σε GPU 12 GB). Σε **4-bit GGUF Q4_K_M** (~0.5 byte/βάρος συν κλίμακες ομάδων) συρρικνώνεται στα **~5.3 GB**, χωρώντας άνετα σε μία **NVIDIA RTX 3060 12 GB** με **~6.7 GB ελεύθερα** για το KV-Cache!

Ένα αποφασιστικό θεωρητικό άλμα συνιστά ο τύπος **NormalFloat4 (NF4, Dettmers et al.)**: αντί τα 16 επίπεδα των 4 bits να απέχουν ισαπέχοντα (ομοιόμορφο πλέγμα), η NF4 τοποθετεί κάθε επίπεδο στα ακριβή *ποσοστημόρια ίσης πιθανότητας* της κανονικής κατανομής *N(0, 1)*. Έτσι συγκεντρώνει πυκνότερη ανάλυση κοντά στο μηδέν, όπου βρίσκεται η συντριπτική πλειονότητα των βαρών, ελαχιστοποιώντας θεωρητικά το σφάλμα ανακατασκευής.

Αντιμετώπιση Ακραίων Καναλιών (Outlier Channels): LLM.int8(), GPTQ, AWQ & GGUF

Σε μοντέλα άνω των ~6.7 δισεκατομμυρίων παραμέτρων εμφανίζεται το συστηματικό φαινόμενο των **Ακραίων Χαρακτηριστικών (Outlier Channels)**: ένα ελάχιστο ποσοστό (~0.1%) των διαστάσεων ενεργοποίησης λαμβάνει τιμές 10× έως 100× μεγαλύτερες από τον μέσο όρο. Εάν κβαντοποιηθεί αφελώς ολόκληρος ο πίνακας, οι ακραίες τιμές διευρύνουν βίαια την κλίμακα *s*, συνθλίβοντας το υπόλοιπο 99.9% των βαρών στο μηδέν. Τέσσερις εμβληματικοί αλγόριθμοι επιλύουν αυτό το πρόβλημα με διαφορετικές μαθηματικές στρατηγικές:

Μέθοδος	Κύρια Μαθηματική Αρχή	Τυπική Χρήση & Πλεονέκτημα
LLM.int8()	Μικτή αποσύνθεση: διατηρεί τα ακραία κανάλια (>6.0) σε FP16 και κβαντοποιεί το 99.9% σε INT8	Μηδενική υποβάθμιση ακρίβειας χωρίς ανάγκη συνόλου βαθμονόμησης
GPTQ	Βελτιστοποίηση 2ης τάξης (αντίστροφος Εσσιανός πίνακας) που αντισταθμίζει το σφάλμα στρογγυλοποίησης στα υπόλοιπα βάρη	Υψηλή συμπίεση 4-bit / 3-bit σε GPU με ταχύτατους πυρήνες ExLlama / Marlin
AWQ (Activation-Aware)	Προστατεύει το ~1% των κρισιμότερων βαρών μέσω κλιμάκωσης ανά κανάλι βάσει του μεγέθους ενεργοποιήσεων	Εξαιρετική γενίκευση εκτός κατανομής, ιδανικό για παραγωγικό vLLM serving
GGUF (Q4_K_M)	Υπερ-μπλοκ μικτής ακρίβειας (k-quants) που δίνουν 5 έως 6 bits στα ευαίσθητα στρώματα προσοχής/FFN και 4 bits στα υπόλοιπα	Καθολικό πρότυπο τοπικής εκτέλεσης (Ollama / llama.cpp) σε CPU, Apple Silicon & GPU



Σχήμα 10.1

Τρισδιάστατη κατάρρευση νέφους βαρών από συνεχή FP16 σε κρυσταλλικό πλέγμα Group-64 INT4 (Q4_K_M): συρρίκνωση μοντέλου 8B στα 5.3 GB VRAM (6.7 GB ελεύθερα σε RTX 3060 12 GB) με 62.4 tok/s και ελάχιστη μεταβολή perplexity (+0.08).

10.3 Το Όριο Εύρους Ζώνης Μνήμης (Memory-Bandwidth Bottleneck)

Γιατί η κβαντοποίηση 4-bit δεν μειώνει απλώς τη μνήμη, αλλά **τριπλασιάζει και την ταχύτητα παραγωγής κειμένου** πάνω στην ίδια GPU; Στην αυτοπαλίνδρομη φάση Decode (με ένα αίτημα χρήστη, $B = 1$), η αριθμητική ένταση (δηλαδή ο λόγος υπολογισμών ανά byte που διαβάζεται από τη μνήμη) είναι ελάχιστη: για να παραχθεί *κάθε ένα νέο token*, η GPU πρέπει να μεταφέρει *ολόκληρο* το μοντέλο από τη μνήμη VRAM στους πυρήνες υπολογισμού. Κατά συνέπεια, το ανώτατο φυσικό όριο ταχύτητας καθορίζεται άμεσα από το εύρος ζώνης μνήμης:

Ταχύτητα Παραγωγής (tok/s) ≈ Εύρος Ζώνης Μνήμης VRAM (GB/s) / Μέγεθος Μοντέλου (GB)

Σε μία NVIDIA RTX 3060 12 GB (BW = 360 GB/s), ένα μοντέλο 8B σε Q4_K_M (~5.3 GB) φτάνει θεωρητικό ταβάνι ~68 tok/s (πραγματικά 55 έως 62 tok/s μετά την ανάγνωση του KV-Cache), ενώ αν υπερχείλιζε στη μνήμη RAM του συστήματος (~40 GB/s) η ταχύτητα θα κατέρρεε κάτω από 4 tok/s!



Σχήμα 10.2

Οικογένειες Open-Weight LLMs, συμπίεση μνήμης VRAM για μοντέλο 8B (FP32, FP16, INT8, INT4 GGUF κάτω από το όριο των 12 GB RTX 3060) και μηχανές τοπικής εκτέλεσης (Ollama, llama.cpp, vLLM, Hugging Face).

10.4 Εργαστήρια 25 & 26: Μαθηματικά INT4 & Τοπικός Πελάτης Ollama

ΕΡΓΑΣΤΗΡΙΟ 25 **Κβαντοποίηση Συμμετρικού INT8 & Ομαδικού Group-64 INT4 με Ακραία Κανάλια**

`code/25_quantization_fp16_to_int4_math.py`

Δοκιμάζει πειραματικά σε ρεαλιστικό πίνακα προβολής Transformer 1.048.576 βαρών (1024 × 1024) με εμφυτευμένα ακραία κανάλια (3.5×) τη διατήρηση του γινομένου $y = Wx$ υπό INT8 ανά γραμμή και **Group-64 INT4** (4.25 bits/βάρος, **Cosine Similarity 0.9896**), ενώ ελέγχει τη χωρητικότητα μοντέλων 4B έως 70B σε GPU 12 GB.

Προτεινόμενος Πειραματισμός: Αλλάξτε το `group_size` σε 32, 128 και 1024 (per-row INT4) και παρατηρήστε πώς οι ακραίες στήλες (`W_fp32[:, ::64] *= 3.5`) υποβαθμίζουν το σφάλμα MSE όταν οι ομάδες γίνονται υπερβολικά μεγάλες.

PYTHON 3.12+ `code/25_quantization_fp16_to_int4_math.py` Εργαστήριο 25: Κβαντοποίηση INT8 & Group-64 INT4 και έλεγχος χωρητικότητας GPU VRAM

```
import numpy as np
```

```python
# === 1. Αλγόριθμοι Κβαντοποίησης INT8 & Ομαδικού INT4 ===

def quantize_symmetric_int8(W: np.ndarray):
    """Εφαρμόζει συμμετρική κβαντοποίηση INT8 ανά γραμμή με κλίμακα max(|W|) / 127."""
    scale = np.max(np.abs(W), axis=-1, keepdims=True) / 127.0 + 1e-12
    q = np.clip(np.round(W / scale), -127, 127).astype(np.int8)
    W_hat = q.astype(np.float32) * scale
    return q, scale, W_hat


def quantize_groupwise_int4(W: np.ndarray, group_size: int = 64):
    """Ομαδική κβαντοποίηση 4-bit (αποθηκεύει 1 κλίμακα FP16 ανά 64 βάρη, ~4.25 bits/βάρος)."""
    orig_shape = W.shape
    flat = W.reshape(-1, group_size)
    scales = np.max(np.abs(flat), axis=1, keepdims=True) / 7.0 + 1e-12
    # Προσημασμένο εύρος 4-bit [-7, +7] ώστε να διατηρείται συμμετρική αναπαράσταση γύρω από το μηδέν
    q4 = np.clip(np.round(flat / scales), -7, 7).astype(np.int8)
    W_hat = (q4.astype(np.float32) * scales).reshape(orig_shape)
    eff_bits = 4.0 + 16.0 / group_size
    return q4, scales, W_hat, eff_bits


def cosine_sim(a: np.ndarray, b: np.ndarray) -> float:
    """Υπολογίζει την Ομοιότητα Συνημιτόνου (Cosine Similarity) μεταξύ δύο διανυσμάτων."""
    return float(np.dot(a.ravel(), b.ravel()) / (np.linalg.norm(a) * np.linalg.norm(b) + 1e-12))


# === 2. Πειραματικός Έλεγχος Πιστότητας & Υπολογιστής Μνήμης GPU VRAM ===

def main():
    """Αξιολογεί το σφάλμα προβολής y = W x υπό INT8/INT4 και ελέγχει τη χωρητικότητα VRAM."""
    rng = np.random.default_rng(2026)

    # Ρεαλιστικός πίνακας προβολής Transformer (1024 x 1024 = 1.048.576 βάρη) με ακραίες στήλες
    W_fp32 = rng.normal(0.0, 0.04, size=(1024, 1024)).astype(np.float32)
    # Προσομοίωση ακραίων καναλιών ενεργοποίησης και βαρών ανά 64 στήλες
    W_fp32[:, ::64] *= 3.5
    W_fp16 = W_fp32.astype(np.float16).astype(np.float32)

    _, _, W_i8 = quantize_symmetric_int8(W_fp16)
    _, _, W_i4, eff_b4 = quantize_groupwise_int4(W_fp16, group_size=64)

    # Έλεγχος πραγματικού γινομένου πίνακα με διάνυσμα y = W x
    x = rng.normal(0.0, 1.0, size=(1024,)).astype(np.float32)
    y_ref = W_fp16 @ x
    y_i8 = W_i8 @ x
    y_i4 = W_i4 @ x

    print("=== 1. WEIGHT QUANTIZATION FIDELITY (1024x1024 LLM LAYER) ===")
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/25_quantization_fp16_to_int4_math.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 26 **Τοπικός Πελάτης Ροής Tokens (Ollama REST API) & Τηλεμετρία tok/s**

`code/26_local_llm_ollama_and_hf.py`

Συνδέεται με μηδενικές εξωτερικές εξαρτήσεις στον τοπικό εξυπηρετητή **Ollama** (`http://localhost:11434/api/generate`), αποκωδικοποιεί τη ροή NDJSON διακριτικό προς διακριτικό και μετρά ζωντανά την πραγματική ταχύτητα `tok/s` (διαθέτοντας ασφαλή εκπαιδευτική προσομοίωση εάν η υπηρεσία δεν τρέχει τοπικά).

Προτεινόμενος Πειραματισμός: Εκτελέστε τοπικά `ollama pull gemma3:4b` και συγκρίνετε την ταχύτητα `tok/s` με το θεωρητικό όριο `BW / Model_GB` της δικής σας GPU.

PYTHON 3.12+ **code/26_local_llm_ollama_and_hf.py** Εργαστήριο 26: Αυτόματος υπολογισμός VRAM & τοπικός πελάτης Ollama (100% offline)

```python
import json
import time
```

```
import urllib.error
import urllib.request


# === 1. Ρύθμιση Τοπικού Τελικού Σημείου Ollama ===

OLLAMA_URL = "http://localhost:11434/api/generate"


# === 2. Κλήση Ροής Tokens & Μέτρηση Διαμεταγωγής ===

def query_ollama_stream(prompt: str, model: str = "gemma3:4b", temperature: float = 0.4):
    """Αποστέλλει προτροπή στον τοπικό διακομιστή Ollama με ροή tokens και καταγράφει τηλεμετρία."""
    payload = json.dumps({
        "model": model,
        "prompt": prompt,
        "stream": True,
        "options": {"temperature": temperature, "num_predict": 140},
    }).encode("utf-8")

    req = urllib.request.Request(OLLAMA_URL, data=payload, headers={"Content-Type": "application/json"})
    t0 = time.perf_counter()
    tokens = 0
    chunks = []

    # Ανάγνωση απάντησης γραμμή προς γραμμή (NDJSON) καθώς παράγεται κάθε νέο token
    with urllib.request.urlopen(req, timeout=8) as resp:
        for raw_line in resp:
            obj = json.loads(raw_line.decode("utf-8"))
            tok = obj.get("response", "")
            if tok:
                print(tok, end="", flush=True)
                chunks.append(tok)
                tokens += 1
            if obj.get("done"):
                break

    dt = max(1e-4, time.perf_counter() - t0)
    print(f"\n\n[Telemetry] Model={model} | Tokens={tokens} | Elapsed={dt:.2f}s | Speed={tokens/dt:.1f}
tok/s")
    return "".join(chunks)


# === 3. Κύρια Ροή Εκτέλεσης & Ασφαλής Προσομοίωση Εκτός Σύνδεσης ===

def main():
    """Δοκιμάζει ζωντανή κλήση στο τοπικό Ollama ή εμφανίζει εκπαιδευτική προσομοίωση Q4_K_M."""
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/26_local_llm_ollama_and_hf.py του code.zip)
```

▸ **Οδηγός Επιλογής Τοπικού Ανοιχτού Μοντέλου για 12 GB GPU (RTX 3060 / 4070)**

1. Καθημερινή Ελληνική & Πολυτροπική Χρήση: `gemma3:4b` ή `gemma3:12b` (Q4_K_M, ~8.1 GB VRAM, εξαιρετική απόδοση στα Ελληνικά). **2. Κώδικας & Μηχανική Λογισμικού:** `qwen2.5-coder:7b` ή `14b` (Q4_K_M, ~8.8 GB VRAM). **3. Βαθύς Μαθηματικός Συλλογισμός:** `deepseek-r1:8b` ή `14b` με 100% εκτέλεση εκτός δικτύου και μηδενική έκθεση ιδιωτικών δεδομένων.

ΚΕΦΑΛΑΙΟ 11

Εξειδίκευση Μοντέλων, LoRA / QLoRA & Ευθυγράμμιση (Alignment)

Υπόθεση χαμηλής τάξης LoRA, διπλή κβαντοποίηση QLoRA και η πλήρης μαθηματική τριάδα ευθυγράμμισης: SFT, RLHF/PPO, κλειστής μορφής DPO και επαληθεύσιμη συλλογιστική GRPO.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Υπόθεση Εγγενούς Χαμηλής Τάξης:** Κατανόηση του γιατί η εξειδίκευση ενός LLM απαιτεί τροποποίηση μόλις 0.39% των παραμέτρων χωρίς καταστροφική λήθη.
- **Μαθηματικά LoRA & QLoRA:** Αρχικοποίηση A (Γκαουσιανή) και $B = 0$, επιλογή τάξης r και κλιμάκωσης α/r, Διπλή Κβαντοποίηση NF4 και Paged Optimizers.
- **Πλήρεις Εξισώσεις Ευθυγράμμισης:** Εμβάθυνση σε SFT Cross-Entropy, RLHF με μοντέλο Bradley-Terry και ποινή KL, κλειστή απώλεια **DPO** και σχετικό πλεονέκτημα ομάδας **GRPO**.
- **Επαλήθευση Συγχώνευσης Μηδενικής Καθυστέρησης:** Υλοποίηση `LoRALinear` από το μηδέν και έλεγχος ακρίβειας μηχανής στο Εργαστήριο 27.

11.1 Υπόθεση Εγγενούς Χαμηλής Τάξης, LoRA & Μηχανική QLoRA

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Το Ριζόχαρτο Πάνω στην Εγκυκλοπαίδεια: Διαίσθηση LoRA, SFT, DPO & GRPO

Φανταστείτε ότι διαθέτετε μια τεράστια τυπωμένη εγκυκλοπαίδεια 50.000 σελίδων (τα προεκπαιδευμένα βάρη $\mathbf{W_0}$ ενός LLM) και θέλετε να την εξειδικεύσετε στη ραδιοαστρονομία ή την ιατρική. Η πλήρης επανεκπαίδευση (**Full Fine-Tuning**) ισοδυναμεί με το να ξανατυπώσετε ολόκληρη την εγκυκλοπαίδεια από την αρχή, με τεράστιο κόστος και κίνδυνο να διαγράψετε παλιές γνώσεις (*καταστροφική λήθη*). Η μέθοδος **LoRA** προσφέρει μια ευφυή εναλλακτική: αφήνει την εγκυκλοπαίδεια ανέπαφη και τοποθετεί επάνω της ένα **λεπτό διαφανές ριζόχαρτο (tracing sheet)** όπου γράφετε μόνο τις στοχευμένες διορθώσεις (`A` και `B`), με μέγεθος λίγων megabytes!

- **SFT (Επιβλεπόμενη Εξειδίκευση):** Σαν ένας φοιτητής που μελετά *λυμένες ασκήσεις* ώστε να μάθει τη σωστή δομή, το ύφος και τη σαφήνεια των απαντήσεων.
- **DPO / RLHF (Ευθυγράμμιση Προτιμήσεων):** Σαν να συγκρίνετε *δύο λύσεις* του ίδιου θέματος (μία εξαιρετική και μία μέτρια) και να μαθαίνετε ακριβώς γιατί υπερέχει η πρώτη.
- **GRPO (Ομαδική Ενίσχυση με Ελεγκτή):** Το μοντέλο γράφει πολλές προσπάθειες για το ίδιο μαθηματικό πρόβλημα, ένας αυτόματος ελεγκτής βαθμολογεί σωστό (1) ή λάθος (0), και ενισχύονται οι διαδρομές σκέψης που ξεπερνούν τον μέσο όρο της ομάδας!

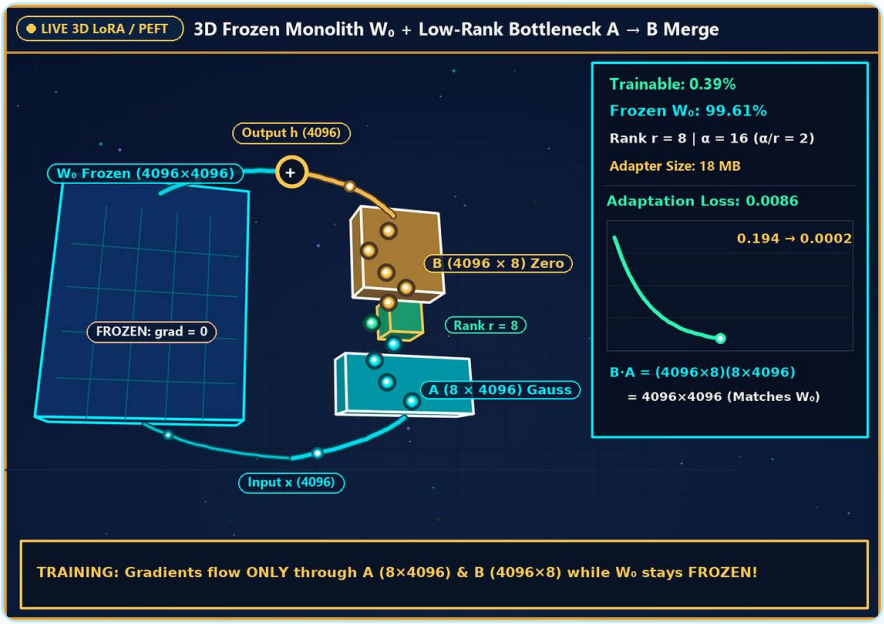
Η πλήρης επανεκπαίδευση (*Full Fine-Tuning*) ενός μοντέλου 8B σε FP16 απαιτεί πάνω από **60 GB GPU VRAM** (16 GB βάρη + 16 GB κλίσεις διόρθωσης *gradients* + 32 GB καταστάσεις ροπής FP32 του βελτιστοποιητή AdamW), ενώ εγκυμονεί τον κίνδυνο *καταστροφικής λήθης (catastrophic forgetting)* της γενικής γνώσης. Θεμελιώδεις μελέτες (**Aghajanyan et al., 2021**) απέδειξαν ότι τα προεκπαιδευμένα γλωσσικά μοντέλα διαθέτουν εξαιρετικά χαμηλή *εγγενή διάσταση (intrinsic dimensionality)*: όταν τα προσαρμόζουμε σε έναν εξειδικευμένο επιστημονικό τομέα, η αναγκαία μεταβολή βαρών ΔW ζει σε έναν συμπαγή υπόχωρο χαμηλής τάξης.

Αξιοποιώντας αυτή την αρχή, η μέθοδος **LoRA (Low-Rank Adaptation, Hu et al., 2021)** παγώνει πλήρως τον προεκπαιδευμένο πίνακα $W_0 \in \mathbb{R}^{d \times d}$ ($\nabla W_0 = 0$) και παραγοντοποιεί τη μεταβολή μέσω δύο λεπτών εκπαιδεύσιμων πινάκων χαμηλής τάξης $r \ll d$: τον **A (r × d)** με Γκαουσιανή αρχικοποίηση $N(0, \sigma^2)$ και τον **B (d × r)** με **μηδενική αρχικοποίηση (B = 0)**. Έτσι στο βήμα $t = 0$ ισχύει ακριβώς $\Delta W = BA = 0$, εξασφαλίζοντας ομαλή εκκίνηση χωρίς διαταραχή των προεκπαιδευμένων αναπαραστάσεων. (Στη σύμβαση γραμμο-διανυσμάτων $X \in \mathbb{R}^{N \times d}$ της PyTorch/NumPy, όπως στο Εργαστήριο 27, η αντίστοιχη ισοδύναμη μορφή είναι $Y = XW_0 + (\alpha/r)(XA)B$.)

$$W' = W_0 + (\alpha / r) \cdot B \cdot A \quad | \quad h = W_0 x + (\alpha / r) \cdot B(Ax)$$

Για d = 4096 και τάξη r = 8, οι εκπαιδεύσιμες παράμετροι πέφτουν από 16.777.216 σε 2 × 4096 × 8 = 65.536 (μόλις 0.39%!). Μετά την εκπαίδευση συγχωνεύουμε αλγεβρικά το (α/r)BA στο W_0 για **μηδενική επιπλέον καθυστέρηση** κατά την εκτέλεση.

- **Οδηγός Επιλογής Υπερπαραμέτρων (r, α & Target Modules):** Η τάξη *r = 4 έως 8* επαρκεί για ύφος, μορφοποίηση και εξειδικευμένη ορολογία, ενώ *r = 16 έως 64* συνιστάται για σύνθετες αλλαγές συλλογισμού ή κώδικα. Ο λόγος κλιμάκωσης ορίζεται τυπικά σε *α = 2r* (ή με σταθεροποιημένη κλιμάκωση *rsLoRA: α/√r* για υψηλά *r*). Η εφαρμογή LoRA σε *όλες* τις γραμμικές προβολές (Q, K, V, O καθώς και τα στρώματα FFN) γεφυρώνει σχεδόν πλήρως την απόδοση με το Full Fine-Tuning.
- **QLoRA (Dettmers et al., 2023):** Συνδυάζει τρεις καινοτομίες ώστε η εκπαίδευση μοντέλων 8B έως 14B να χωρά σε μία μόνο GPU 12 GB: (α) παγωμένη βάση W_0 σε **4-bit NF4** (αποκβαντοποιείται δυναμικά σε BF16 μόνο κατά τον πολλαπλασιασμό), (β) **Διπλή Κβαντοποίηση (Double Quantization)** που κβαντοποιεί και τους ίδιους τους συντελεστές κλίμακας από FP32 σε FP8 εξοικονομώντας ~0.37 bits ανά παράμετρο, και (γ) **Paged Optimizers** που αξιοποιούν την Ενιαία Μνήμη NVIDIA για αυτόματη μεταφορά καταστάσεων AdamW στη RAM κατά τη διάρκεια στιγμιαίων αιχμών κλίσης.



Σχήμα 11.1

Τρισδιάστατη γεωμετρία προσαρμογέα LoRA: ο παγωμένος μονόλιθος W0 (4096×4096, 99.61% βαρών, grad = 0) παραμένει ανέπαφος, ενώ οι κλίσεις ρέουν αποκλειστικά μέσω του λαιμού χαμηλής τάξης A (8×4096) και B (4096×8) (0.39% βαρών, μέγεθος adapter ~18 MB).

11.2 Πλήρη Μαθηματικά Ευθυγράμμισης (Alignment): SFT, RLHF, DPO & GRPO

Ένα προεκπαιδευμένο βασικό μοντέλο (*Base Model*) γνωρίζει τεράστιο όγκο πληροφοριών αλλά δεν ακολουθεί οδηγίες ούτε ελέγχει την εγκυρότητα της σκέψης του. Η μετατροπή του σε αξιόπιστο επιστημονικό συνεργάτη επιτυγχάνεται μέσα από την τριάδα μετεκπαίδευσης (*Post-Training*):

1. **Επιβλεπόμενη Εξειδίκευση Οδηγιών (SFT):** Εκπαίδευση με τυπική απώλεια διασταυρούμενης εντροπίας (*cross-entropy*) υπολογισμένη αποκλειστικά πάνω στα διακριτικά της ιδανικής απόκρισης *y* δοθείσας της προτροπής *x*: $L_{SFT}(\theta) = -\Sigma_t \log \pi_\theta(y_t \mid x, y_{<t})$. Εκπαιδεύεται μόνο ένα μοντέλο στη VRAM (η πολιτική π_θ).
2. **Ενισχυτική Μάθηση από Ανθρώπινη Ανατροφοδότηση (RLHF + PPO):** Πρώτα εκπαιδεύεται ένα νευρωνικό **Μοντέλο Ανταμοιβής (Reward Model, r_ϕ)** πάνω σε ζεύγη ανθρώπινων προτιμήσεων (προτιμητέα y_w έναντι απορριφθείσας y_l) με το πιθανοτικό μοντέλο **Bradley-Terry**: $P(y_w > y_1 \mid x) = \sigma(r_\phi(x, y_w) - r_\phi(x, y_1))$. Κατόπιν η πολιτική βελτιστοποιείται με PPO μεγιστοποιώντας την ανταμοιβή υπό ποινή απόκλισης KL από το παγωμένο μοντέλο αναφοράς π_{ref}: $\max E[r_\phi(x, y)] - \beta D_{KL}(\pi_\theta \,||\, \pi_{ref})$. Η ποινή β D_{KL} αποτρέπει την κατάρρευση της

γλωσσικής ποιότητας (*reward hacking*), όμως η PPO απαιτεί **4 μοντέλα ταυτόχρονα στη VRAM** (Policy, Reference, Reward, Critic).

3. **Άμεση Βελτιστοποίηση Προτιμήσεων (DPO, Rafailov et al., 2023):** Αποδεικνύει αναλυτικά ότι η βέλτιστη αντιμοιβή συνδέεται κλειστά με τον λογάριθμο του λόγου πιθανοτήτων π_θ / π_{ref}, καταργώντας *πλήρως* το Reward Model και τον Critic (μόνο 2 μοντέλα στη VRAM!) με μία απλή, σταθερή συνάρτηση απώλειας κλειστής μορφής.
4. **Ομαδική Σχετική Βελτιστοποίηση Πολιτικής (GRPO, DeepSeek-R1):** Για πεδία με **αντικειμενικά επαληθεύσιμη αλήθεια** (μαθηματικές αποδείξεις, μεταγλώττιση και unit tests κώδικα, φυσικές προσομοιώσεις), το μοντέλο παράγει μια ομάδα G υποψήφιων απαντήσεων $\{o_1, \ldots, o_G\}$ ανά πρόβλημα, κάθε απάντηση βαθμολογείται από ντετερμινιστικό ελεγκτή ($r_i \in \{0, 1\}$) και το **Σχετικό Πλεονέκτημα Ομάδας (A_i)** ενισχύει τις σωστές διαδρομές χωρίς καθόλου νευρωνικό Critic! Αυτή η τεχνική (*RLVR*) γεννά αυθόρμητα συμπεριφορές αυτοδιόρθωσης και βαθύ συλλογισμό.

$$L_{DPO}(\theta) = -E_{(x,\, y_w,\, y_l)} \left[\log \sigma\left(\beta \log(\pi_\theta(y_w|x) / \pi_{ref}(y_w|x)) - \beta \log(\pi_\theta(y_l|x) / \pi_{ref}(y_l|x)) \right) \right]$$

$$\text{GRPO Group Advantage: } A_i = \left(r_i - \text{mean}(r_{1..G}) \right) / \left(\text{std}(r_{1..G}) + \epsilon \right)$$

Η σύγχρονη συνταγή αιχμής συνδυάζει Pre-Training σε τρισεκατομμύρια tokens, SFT + DPO για καθαρό ύφος και ασφάλεια, και GRPO με επαληθεύσιμες ανταμοιβές για επιστημονική και αλγοριθμική ακρίβεια.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Συμπίεση Παραμέτρων LoRA (r = 8) & Ομαδικό Πλεονέκτημα GRPO (4 Απαντήσεις)

Μέρος Α (Γιατί η LoRA εκπαιδεύει 256× λιγότερα βάρη): Ένα τυπικό γραμμικό στρώμα Transformer μεγέθους `4096 × 4096` περιέχει **16.777.216** παγωμένα βάρη W_0. Με προσαρμογέα LoRA τάξης `r = 8`, προσθέτουμε μόνο δύο λεπτούς πίνακες: **A (8 × 4096) = 32.768** και **B (4096 × 8) = 32.768**. Σύνολο εκπαιδεύσιμων παραμέτρων: **65.536**, δηλαδή *65.536 / 16.777.216* = ***0.39%*** (**256×** λιγότερα βάρη και κλίσεις στη VRAM!).

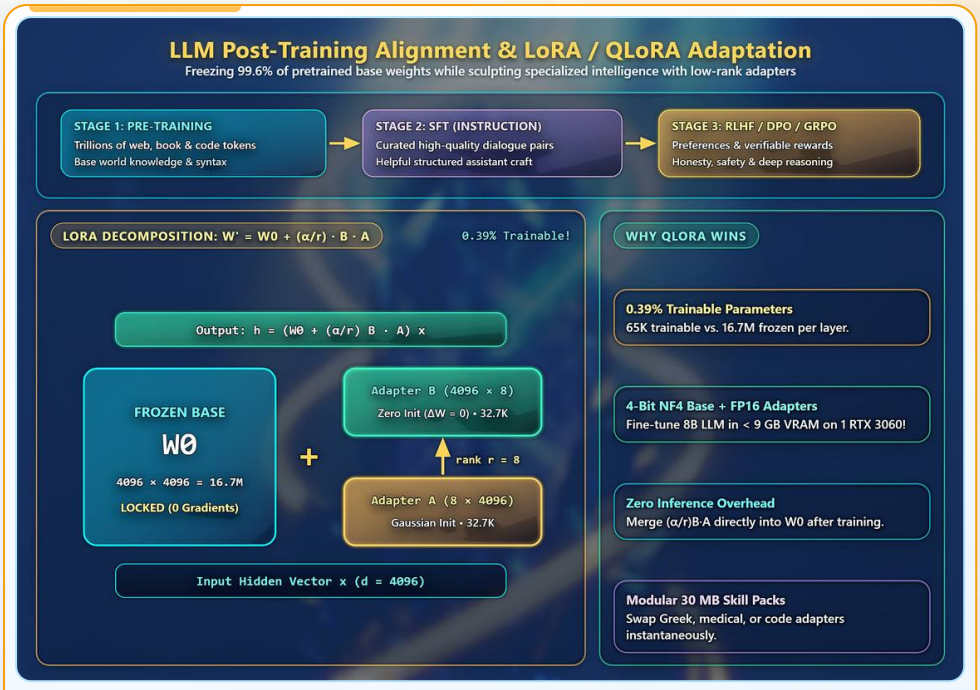
Μέρος Β (Ομαδικό Πλεονέκτημα GRPO χωρίς νευρωνικό Critic): Έστω ότι το μοντέλο παράγει $G = 4$ ανεξάρτητες λύσεις για μία εξίσωση και ο αυτόματος ελεγκτής αποδίδει δυαδικές ανταμοιβές `r = [1, 0, 1, 0]` (δύο σωστές, δύο λάθος):

1. **Μέσος Όρος Ομάδας:** $\mu = (1 + 0 + 1 + 0) / 4 = \mathbf{0.5}$.
2. **Τυπική Απόκλιση:** $\sigma = \sqrt{[(0.5^2 + (-0.5)^2 + 0.5^2 + (-0.5)^2) / 4]} = \sqrt{0.25} = \mathbf{0.5}$.
3. **Κανονικοποιημένο Πλεονέκτημα $A_i = (r_i - \mu) / \sigma$:** Για $r_i = 1 \rightarrow (1 - 0.5)/0.5 = \mathbf{+1.0}$ (ενίσχυση), ενώ για $r_i = 0 \rightarrow (0 - 0.5)/0.5 = \mathbf{-1.0}$ (καταστολή), δίνοντας το καθαρό διάνυσμα `A = [+1, -1, +1, -1]` χωρίς να απαιτείται καθόλου ξεχωριστό δίκτυο Critic!



Σχήμα 11.2

Πλήρης σύγκριση της τριάδας ευθυγράμμισης LLMs: SFT (1 μοντέλο στη VRAM), RLHF + PPO (4 μοντέλα), κλειστής μορφής DPO (2 μοντέλα, χωρίς Reward Model) και επαληθεύσιμη συλλογιστική GRPO (ομαδικό πλεονέκτημα A_i χωρίς Critic).



Σχήμα 11.3

Ενοποιημένος αγωγός παραμετρικά αποδοτικής εξειδίκευσης PEFT / LoRA & QLoRA: πάγωμα του 99.6% των θεμελιωδών βαρών, εκπαίδευση σε 12 GB GPU και αλγεβρική συγχώνευση μηδενικού κόστους εκτέλεσης.

11.3 Εργαστήριο 27: Υλοποίηση LoRA από το Μηδέν & Επαλήθευση Συγχώνευσης

ΕΡΓΑΣΤΗΡΙΟ 27

Γραμμικό Στρώμα LoRA (r = 8) από το Μηδέν & Επαλήθευση Συγχώνευσης Μηδενικού Κόστους

`code/27_lora_from_scratch_pytorch.py`

Κατασκευάζει τη διαφανή κλάση `LoRALinearNumPy` χωρίς κρυμμένα πλαίσια: παγώνει τον πίνακα W_0 (256 × 256), αρχικοποιεί τον $B = 0$, υπολογίζει αναλυτικά τις κλίσεις `grad_A` και `grad_B` ρίχνοντας την απώλεια προσαρμογής από **0.1936 σε 0.000000** και επαληθεύει ότι η συγχώνευση `merge_weights()` δίνει ταυτόσημη έξοδο με ακρίβεια μηχανής **~10^{-15}**.

Προτεινόμενος Πειραματισμός: Δοκιμάστε `rank=2`, `rank=4` και `rank=16` (με τον πραγματικό στόχο τάξης 4 στο σενάριο) και παρατηρήστε γιατί `rank < 4` αφήνει μη μηδενικό υπόλοιπο απώλειας ενώ `rank ≥ 4` πετυχαίνει ακριβή μηδενική σύγκλιση!

PYTHON 3.12+ `code/27_lora_from_scratch_pytorch.py` Εργαστήριο 27: Εκπαίδευση προσαρμογέων LoRA (r=8) & επαλήθευση συγχώνευσης μηδενικού κόστους

```python
import numpy as np


# === 1. Υλοποίηση Γραμμικού Επιπέδου LoRA σε Καθαρή NumPy ===

class LoRALinearNumPy:
    """Διαφανής μαθηματική υλοποίηση γραμμικού επιπέδου LoRA (Hu κ.ά., 2021) σε NumPy."""

    def __init__(self, d_in: int, d_out: int, rank: int = 8, alpha: float = 16.0, seed: int = 42):
        """Αρχικοποιεί τον παγωμένο πίνακα W0, τον τυχαίο A και τον μηδενικό B (DeltaW = 0)."""
        rng = np.random.default_rng(seed)
        # Παγωμένος βασικός πίνακας προεκπαιδευμένων βαρών W0 (δεν ενημερώνεται κατά την εκπαίδευση)
        self.W0 = rng.normal(0.0, 0.15, size=(d_in, d_out)).astype(np.float64)
        # Εκπαιδεύσιμος πίνακας προβολής σε υποχώρο χαμηλής τάξης r
        self.A = rng.normal(0.0, 0.05, size=(d_in, rank)).astype(np.float64)
        # Μηδενική αρχικοποίηση του B ώστε στο βήμα 0 να ισχύει ακριβώς DeltaW = A @ B = 0
        self.B = np.zeros((rank, d_out), dtype=np.float64)
```

```
        self.scale = alpha / float(rank)

    def forward(self, X: np.ndarray) -> np.ndarray:
        """Υπολογίζει το εμπρόσθιο πέρασμα Y = X @ W0 + (alpha / r) * (X @ A) @ B."""
        return X @ self.W0 + self.scale * ((X @ self.A) @ self.B)

    def train_step(self, X: np.ndarray, Y_target: np.ndarray, lr: float = 0.08) -> float:
        """Εκτελεί ένα βήμα καθόδου κλίσης MSE ενημερώνοντας αποκλειστικά τους A και B."""
        H = X @ self.A
        Y_pred = X @ self.W0 + self.scale * (H @ self.B)
        err = (2.0 / float(Y_target.size)) * (Y_pred - Y_target)
        loss = float(np.mean((Y_pred - Y_target) ** 2))

        # Υπολογισμός αναλυτικών κλίσεων μόνο για τους A και B (ο W0 παραμένει 100% παγωμένος)
        grad_B = self.scale * (H.T @ err)
        grad_A = self.scale * (X.T @ (err @ self.B.T))
        self.A -= lr * grad_A
        self.B -= lr * grad_B
        return loss

    def merge_weights(self) -> np.ndarray:
        """Συγχωνεύει αλγεβρικά τον προσαρμογέα LoRA στον W0 για μηδενική επιπλέον καθυστέρηση."""
        return self.W0 + self.scale * (self.A @ self.B)


# === 2. Πείραμα Προσαρμογής Πεδίου & Επαλήθευση Συγχώνευσης ===

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/27_lora_from_scratch_pytorch.py του code.zip)
```

ΚΕΦΑΛΑΙΟ 12

Συστήματα RAG, Διανυσματικές Βάσεις & Αυτόνομοι Πράκτορες AI (MCP)

Υβριδική ανάκτηση (Dense Embeddings + BM25 + RRF), ευρετήρια HNSW, Cross-Encoder Reranking, βρόχος ReAct και αρχιτεκτονική Client-Host-Server του Model Context Protocol (MCP).

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Υβριδική Ανάκτηση (Dense + BM25 + RRF):** Συνδυασμός σημασιολογικών ενσωματώσεων και ακριβούς λεξιλογικής αναζήτησης μέσω Reciprocal Rank Fusion για εξάλειψη ψευδαισθήσεων.
- **Ευρετηρίαση HNSW, Chunking & Reranking:** Κατανόηση της λογαριθμικής αναζήτησης ANN, των συμβιβασμών τεμαχισμού (overlap) και της επανακατάταξης Cross-Encoder.
- **Αρχιτεκτονική Πρωτοκόλλου MCP:** Σαφής διαχωρισμός Client, Host και Server (JSON-RPC 2.0) και των δομικών στοιχείων Tools, Resources, Prompts και Sandboxed Permissions.
- **Εφαρμοσμένη Μηχανική RAG & ReAct:** Κατασκευή διανυσματικής μηχανής με παραπομπές και αυτόνομου πράκτορα εργαλείων στα Εργαστήρια 28 & 29.

12.1 Πυκνή Ανάκτηση, Αραιή BM25, Υβριδική Σύντηξη RRF & Ευρετήρια HNSW

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ **Εξέταση με Ανοιχτά Βιβλία & Η Θύρα USB-C της AI: Διαίσθηση RAG & MCP**

Για να κατανοήσετε αμέσως τη διαφορά ανάμεσα σε ένα απλό LLM, ένα σύστημα **RAG** και έναν πράκτορα **MCP**, σκεφτείτε τρία απλά σενάρια: ένα τυπικό LLM δίνει **εξέταση με κλειστά βιβλία**, απαντώντας αποκλειστικά από τη μνήμη της εκπαίδευσής του (με κίνδυνο «ψευδαισθήσεων» σε νέα ή ιδιωτικά δεδομένα). Το **RAG (Retrieval-Augmented Generation)** μετατρέπει το μοντέλο σε εξεταζόμενο με **ανοιχτά βιβλία**: πρώτα εντοπίζει τις ακριβείς σελίδες των εγγράφων σας και έπειτα συνθέτει την απάντηση παραπέμποντας σε αυτές (`[1][2]`). Τέλος, το **Model Context Protocol (MCP)** λειτουργεί σαν την **καθολική θύρα USB-C** της AI, συνδέοντας με ασφάλεια το μοντέλο σε πραγματικά εργαλεία (αριθμομηχανές, βάσεις SQL, κώδικα Python) ώστε να εκτελεί ακριβείς πράξεις αντί να μαντεύει!

- **Embedding (Ενσωμάτωση):** Ένα αριθμητικό διάνυσμα-«δακτυλικό αποτύπωμα» που τοποθετεί προτάσεις με παρόμοιο νόημα κοντά στον γεωμετρικό χώρο.
- **BM25 & RRF:** Το **BM25** αναζητά ακριβείς λέξεις-κλειδιά και κωδικούς, ενώ η **RRF (Reciprocal Rank Fusion)** παντρεύει δίκαια τις κατατάξεις της σημασιολογικής και της λεξιλογικής αναζήτησης.
- **Βρόχος ReAct:** Ο ελεγχόμενος κύκλος *Σκέφτομαι → Καλώ Εργαλείο → Διαβάζω το Αποτέλεσμα → Απαντώ Τεκμηριωμένα*.

Ακόμη και το ισχυρότερο γλωσσικό μοντέλο διαθέτει στατική ημερομηνία γνώσης και δεν έχει πρόσβαση στα ιδιωτικά έγγραφα, τα πειραματικά πρωτόκολλα ή τα τρέχοντα δεδομένα ενός εργαστηρίου. Η αρχιτεκτονική **RAG (Retrieval-Augmented Generation)** επιλύει ριζικά αυτό το πρόβλημα: τεμαχίζει ζωντανά τεκμήρια σε σημασιολογικά αποσπάσματα (*chunks*), τα ευρετηριάζει και ενσωματώνει δυναμικά τα σχετικότερα αποσπάσματα στην προτροπή του LLM, παράγοντας απαντήσεις με ακριβείς, ελέγξιμες παραπομπές (`[1][2]`) χωρίς καμία επανεκπαίδευση βαρών.

Στα σύγχρονα συστήματα παραγωγής, η απλή διανυσματική αναζήτηση δεν αρκεί μόνη της: τα **πυκνά νευρωνικά διανύσματα (Dense Embeddings)** κατανοούν άριστα συνώνυμα και εννοιολογική εγγύτητα (π.χ. «συμπίεση 4-bit» ↔ «κβαντοποίηση»), αλλά συχνά αστοχούν σε ακριβείς κωδικούς, ακρωνύμια ή αριθμητικά ονόματα (όπως `FIPS 203`, `blc1` ή `RTX 3060`). Αντίστροφα, η κλασική **αραιή λεξιλογική ανάκτηση BM25** εντοπίζει με απόλυτη ακρίβεια σπάνιους όρους αλλά αγνοεί τη σημασιολογία. Η λύση αιχμής είναι η **Υβριδική Ανάκτηση (Hybrid Retrieval)**, η οποία ενοποιεί τις κατατάξεις των δύο μηχανών μέσω της αμερόληπτης **Αμοιβαίας Σύντηξης Κατάταξης (Reciprocal Rank Fusion, RRF)**:

$$\cos(q, d) = (q \cdot d) / (||q|| \cdot ||d||) \quad | \quad RRF(d) = \Sigma_{i \in \{Dense,\ BM25\}} 1 / (k + rank_i(d)) \quad (k = 60)$$

Με μοναδιαία κανονικοποίηση ($||v|| = 1$), η ομοιότητα συνημιτόνου γίνεται ένα απλό γινόμενο πινάκων (matrix @ q_vec). Η σύντηξη RRF δεν απαιτεί βαθμονόμηση ασύμβατων κλιμάκων βαθμολογίας και αναδεικνύει τεκμήρια που κατατάσσονται ψηλά και στις δύο μεθόδους.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Υπολογισμός Ομοιότητας Συνημιτόνου & Υβριδικής Βαθμολογίας RRF με το Χέρι

Έστω ένα κανονικοποιημένο διάνυσμα ερωτήματος `q = [0.8, 0.6, 0.0]` (μήκος $||q|| = \sqrt{0.64 + 0.36} = 1.0$) και δύο έγγραφα στη διανυσματική βάση: d_1 `= [0.6, 0.8, 0.0]` (σχετικό με GPU/VRAM) και d_2 `= [0.0, 0.0, 1.0]` (άσχετο, π.χ. συνταγή μαγειρικής):

1. **Ομοιότητα Συνημιτόνου με d_1:** Επειδή $||d_1|| = 1.0$, υπολογίζουμε απλώς το εσωτερικό γινόμενο: $cos(q, d_1) = (0.8 \times 0.6) + (0.6 \times 0.8) + (0.0 \times 0.0) = 0.48 + 0.48 = \mathbf{0.96}$ (πολύ ισχυρή σημασιολογική αντιστοίχιση, γωνία ~16.3°!).
2. **Ομοιότητα Συνημιτόνου με d_2:** $cos(q, d_2) = (0.8 \times 0) + (0.6 \times 0) + (0.0 \times 1) = \mathbf{0.00}$ (κάθετα διανύσματα 90°, απολύτως άσχετο έγγραφο, απορρίπτεται!).
3. **Υβριδική Σύντηξη RRF (`k = 60`):** Εάν το d_1 βγει **1ο** στη διανυσματική αναζήτηση (Dense) και **2ο** στη λεξιλογική αναζήτηση BM25, η συνολική του βαθμολογία είναι $RRF(d_1) = 1/(60 + 1) + 1/(60 + 2) = 1/61 + 1/62 = 0.01639 + 0.01613 = \mathbf{0.0325}$. Έτσι ξεπερνά καθαρά ένα έγγραφο που έπιασε απλώς μία λέξη-κλειδί (1ο στο BM25 αλλά 20ό στο Dense: $1/61 + 1/80 = 0.0289$), διασφαλίζοντας κορυφαία ακρίβεια!

- **Ευρετηρίαση Προσεγγιστικών Πλησιέστερων Γειτόνων HNSW (Hierarchical Navigable Small World):** Αντί για γραμμική σάρωση `O(N)` σε εκατομμύρια διανύσματα, το HNSW οργανώνει τα διανύσματα σε ιεραρχικό γράφο πολλαπλών επιπέδων (όπως οι αεροπορικοί κόμβοι μακρινών πτήσεων πάνω από τοπικές οδούς), επιτυγχάνοντας αναζήτηση `O(log N)` σε λίγα χιλιοστά του δευτερολέπτου με ανάκληση >97%.
- **Σημασιολογικός Τεμαχισμός (Semantic Chunking) & Επικάλυψη (Overlap):** Μικρά τεμάχια (200 έως 400 tokens) προσφέρουν οξεία ακρίβεια ανάκτησης, ενώ μεγαλύτερα διατηρούν τη ροή του επιχειρήματος. Επικάλυψη 10% έως 20% (ή ιεραρχική ανάκτηση *Parent-Child*) αποτρέπει τη βίαιη αποκοπή εξισώσεων και πινάκων στα όρια των τεμαχίων.
- **Επανακατάταξη Cross-Encoder & Αξιολόγηση Πιστότητας (Faithfulness):** Αφού ο γρήγορος *Bi-Encoder* (που κωδικοποιεί χωριστά ερώτημα και έγγραφο) ανακτήσει τα κορυφαία 30 έως 50 υποψήφια αποσπάσματα, ένας **Cross-Encoder** επεξεργάζεται από κοινού το ερώτημα και κάθε απόσπασμα με πλήρη διασταυρούμενη προσοχή, φιλτράροντας τα 5 ακριβέστερα τεκμήρια. Η ποιότητα του συστήματος μετριέται αυστηρά σε τρεις άξονες: *Context Recall* (βρέθηκε η τεκμηρίωση;), *Faithfulness* (στηρίζεται κάθε ισχυρισμός της απάντησης αποκλειστικά στα ανακτημένα αποσπάσματα;) και *Answer Relevance*.

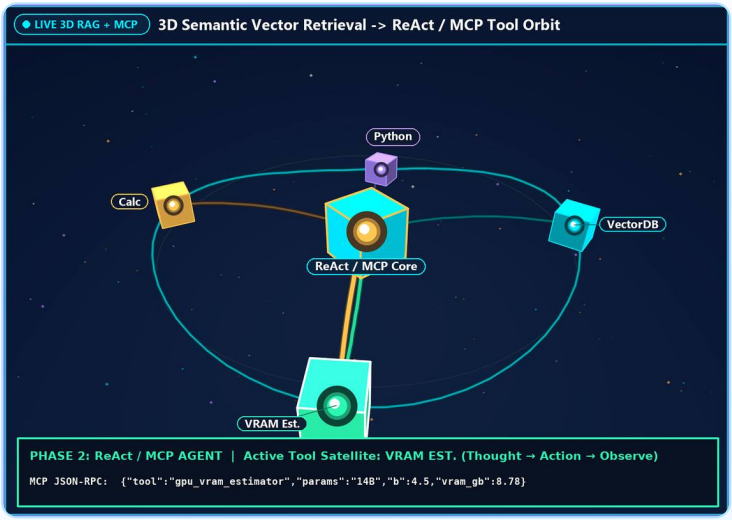
12.2 Αυτόνομοι Πράκτορες ReAct & Αρχιτεκτονική Model Context Protocol (MCP)

Οι σύγχρονοι **Αυτόνομοι Πράκτορες AI (AI Agents)** υπερβαίνουν την παθητική παραγωγή κειμένου μέσω του επαναληπτικού βρόχου **ReAct (Reason + Act)**: *Σκέψη (Thought) → Δομημένη Κλήση Εργαλείου JSON (Tool Call) → Ντετερμινιστική Παρατήρηση (Observation) → Επαληθευμένη Τελική Απάντηση*. Αντί το LLM να «μαντεύει» υπολογισμούς VRAM, SQL ερωτήματα ή φυσικές εξισώσεις, αναθέτει την εκτέλεση σε ελεγχόμενα ντετερμινιστικά εργαλεία.

Το ανοιχτό πρότυπο **Model Context Protocol (MCP)** καθιερώνει μια τυποποιημένη αρχιτεκτονική **Client-Host-Server** πάνω σε **JSON-RPC 2.0** (τοπικά μέσω `stdio` ή απομακρυσμένα μέσω Streamable HTTP), λειτουργώντας ως καθολική διεπαφή «USB-C για εργαλεία AI» που διαχωρίζει με σαφήνεια τα δικαιώματα και τις αρμοδιότητες:

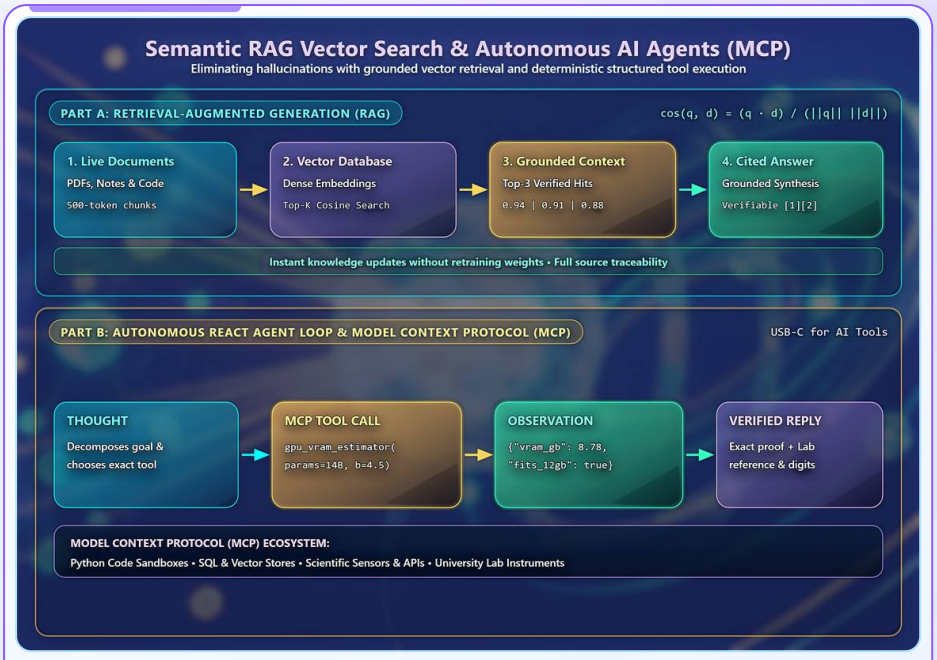
Συνιστώσα / Δομικό Στοιχείο MCP	Φορέας Ελέγχου & Ρόλος (JSON-RPC 2.0)	Επιστημονικό & Μηχανικό Παράδειγμα
Host, Client & Server	Το *Host* (IDE/εφαρμογή LLM) διαχειρίζεται το πλαίσιο και τη συναίνεση χρήστη, το *Client* διατηρεί 1:1 σύνδεση με κάθε απομονωμένο *Server*	Ένα πανεπιστημιακό περιβάλλον AI συνδέεται ταυτόχρονα με τοπικό Python Server, βάση SQL και κατάλογο δεδομένων

Συνιστώσα / Δομικό Στοιχείο MCP	Φορέας Ελέγχου & Ρόλος (JSON-RPC 2.0)	Επιστημονικό & Μηχανικό Παράδειγμα
Tools (Εργαλεία)	Ελεγχόμενα από το μοντέλο (*Model-controlled*): αυστηρά σχήματα JSON εισόδου/εξόδου για ντετερμινιστική εκτέλεση	`gpu_vram_estimator()`, ασφαλής μαθηματικός υπολογιστής, έλεγχος RFI ραδιοτηλεσκοπίου
Resources (Πόροι)	Ελεγχόμενοι από την εφαρμογή (*Application-controlled*): αναγνώσιμο περιεχόμενο με διευθύνσεις URI	Σχήματα πινάκων βάσης δεδομένων, αρχεία καταγραφής πειραμάτων, ζωντανή τηλεμετρία αισθητήρων
Prompts (Προτροπές)	Ελεγχόμενες από τον χρήστη (*User-controlled*): επαναχρησιμοποιήσιμα πρότυπα ροής εργασίας	Τυποποιημένος έλεγχος ποιότητας κώδικα, πρωτόκολλο διαλογής αστρονομικού σήματος
Sandboxed Permissions	Αρχή ελάχιστων προνομίων, επικύρωση σχήματος, ανθρώπινη έγκριση για μεταβολές κατάστασης & άμυνα κατά *Prompt Injection*	Εκτέλεση κώδικα χωρίς πρόσβαση στο σύστημα αρχείων και απαίτηση επιβεβαίωσης πριν από κάθε εγγραφή



Σχήμα 12.1

Τρισδιάστατη τροχιά πράκτορα ReAct / MCP: ο κεντρικός πυρήνας συλλογισμού καλεί μέσω δομημένου JSON-RPC ντετερμινιστικούς δορυφόρους εργαλείων (VectorDB, Python Sandbox, Calculator, VRAM Estimator) εκμηδενίζοντας αριθμητικές ψευδαισθήσεις.



Σχήμα 12.2

Αρχιτεκτονική σημασιολογικής ανάκτησης RAG (έγγραφα, διανυσματική βάση, Top-K συνημίτονο, τεκμηριωμένες παραπομπές) και κύκλος αυτόνομου πράκτορα ReAct με εργαλεία Model Context Protocol (MCP).

12.3 Εργαστήρια 28 & 29: Διανυσματική Μηχανή RAG & Πράκτορας ReAct/MCP

ΕΡΓΑΣΤΗΡΙΟ 28 **Διανυσματική Βάση RAG από το Μηδέν, TF-IDF L2 & Τεκμηριωμένες Παραπομπές**

`code/28_rag_vector_search_from_scratch.py`

Υλοποιεί σε καθαρή NumPy την κλάση `VectorRAGStore`: ευρετηριάζει τη βάση γνώσης του σεμιναρίου σε κανονικοποιημένα διανύσματα TF-IDF μοναδιαίας νόρμας L2, υπολογίζει ακαριαία την Ομοιότητα Συνημιτόνου μέσω ενός εσωτερικού γινομένου πίνακα-διανύσματος (`self.matrix @ q_vec`) και συνθέτει τεκμηριωμένη προτροπή με ρητές παραπομπές `[1]`, `[2]`, `[3]`.

Προτεινόμενος Πειραματισμός: Προσθέστε δύο νέα αποσπάσματα από τα δικά σας μαθήματα στη λίστα `KNOWLEDGE_BASE` και διατυπώστε ένα ερώτημα με συνώνυμα για να διαπιστώσετε πού πλεονεκτούν οι πυκνές νευρωνικές ενσωματώσεις σε συνδυασμό με υβριδική σύντηξη RRF.

PYTHON 3.12+ `code/28_rag_vector_search_from_scratch.py` Εργαστήριο 28: Διανυσματική βάση RAG & ανάκτηση Top-K με ομοιότητα συνημιτόνου

```python
import math
import re
import numpy as np


# === 1. Τεκμηριωμένη Βάση Γνώσης Σεμιναρίου ===

KNOWLEDGE_BASE = [
    ("Ch.03 | XOR MLP", "A single linear perceptron fails on XOR, whereas a 2-layer MLP with ReLU hidden units bends space to achieve 100 percent accuracy."),
    ("Ch.05 | Transformers", "Scaled dot-product Self-Attention computes softmax of Q times K transpose divided by square root of dk to link distant words globally."),
    ("Ch.08 | GPU CUDA", "GPUs pack thousands of parallel ALUs and Tensor Cores to multiply massive neural matrices 20 to 60 times faster than CPUs."),
    ("Ch.09 | KV-Cache", "During autoregressive LLM decoding, the KV-Cache stores past Key and Value tensors so generation scales linearly instead of quadratically."),
    ("Ch.10 | Quantization", "4-bit GGUF Q4_K_M quantization shrinks an 8B open-weight Llama or Gemma model from 16 GB FP16 down to 4.8 GB to run locally on an RTX 3060."),
    ("Ch.11 | LoRA PEFT", "LoRA freezes 99.4 percent of foundation weights W0 and trains small low-rank matrices A and B so fine-tuning runs on a single university GPU."),
```

```
    ("Ch.12 | MCP Agents", "Model Context Protocol (MCP) standardizes how autonomous AI agents safely discover and call external tools, databases, and code sandboxes."),
    ("Ch.16 | Quantum AI", "A qubit exists on the 3D Bloch sphere in superposition; 300 entangled qubits span more basis states than atoms in the observable universe."),
]


# === 2. Διαχωρισμός Όρων & Διανυσματική Αποθήκη TF-IDF ===

def tokenize(text: str) -> list[str]:
    """Εξάγει αλφαριθμητικούς όρους σε πεζά γράμματα για ευρετηρίαση στον χώρο TF-IDF."""
    return re.findall(r"[a-z0-9]+", text.lower())


class VectorRAGStore:
    """Διανυσματική αποθήκη γνώσης με στάθμιση TF-IDF και αναζήτηση Ομοιότητας Συνημιτόνου."""

    def __init__(self, docs: list[tuple[str, str]]):
        """Δημιουργεί λεξιλόγιο, υπολογίζει βάρη IDF και προ-ενσωματώνει όλα τα τεκμήρια."""
        self.docs = docs
        tokenized = [tokenize(title + " " + body) for title, body in docs]
        self.vocab = sorted({w for toks in tokenized for w in toks})
        self.w2i = {w: i for i, w in enumerate(self.vocab)}

        # Συχνότητα τεκμηρίων (DF) και εξομαλυμένη αντίστροφη συχνότητα τεκμηρίων (IDF)
        df = np.zeros(len(self.vocab), dtype=np.float64)
        for toks in tokenized:
            for w in set(toks):
                df[self.w2i[w]] += 1.0
        self.idf = np.log((len(docs) + 1.0) / (df + 1.0)) + 1.0
        self.matrix = np.vstack([self._embed(title + " " + body) for title, body in docs])

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/28_rag_vector_search_from_scratch.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 29 **Αυτόνομος Πράκτορας ReAct & Μητρώο Ασφαλών Εργαλείων MCP (JSON Tool Calling)**

`code/29_ai_agent_tool_calling_react.py`

Επιδεικνύει τον πλήρη πολυβηματικό βρόχο **Σκέψη → Κλήση JSON → Παρατήρηση → Επαληθευμένη Απάντηση** με μητρώο τριών απομονωμένων εργαλείων MCP: υπολογιστή μνήμης VRAM (`tool_gpu_vram_estimator`), ασφαλή μαθηματικό αποτιμητή χωρίς ενσωματωμένα builtins (`tool_calculator`) και ευρετήριο ύλης (`tool_search_syllabus`).

Προτεινόμενος Πειραματισμός: Προσθέστε ένα τέταρτο εργαλείο `tool_kv_cache_estimator(seq_len, n_kv_heads)` στο `TOOL_REGISTRY` και επεκτείνετε το πλάνο του πράκτορα ώστε να ελέγχει ταυτόχρονα μέγεθος βαρών και μνήμη πλαισίου 32K στην RTX 3060.

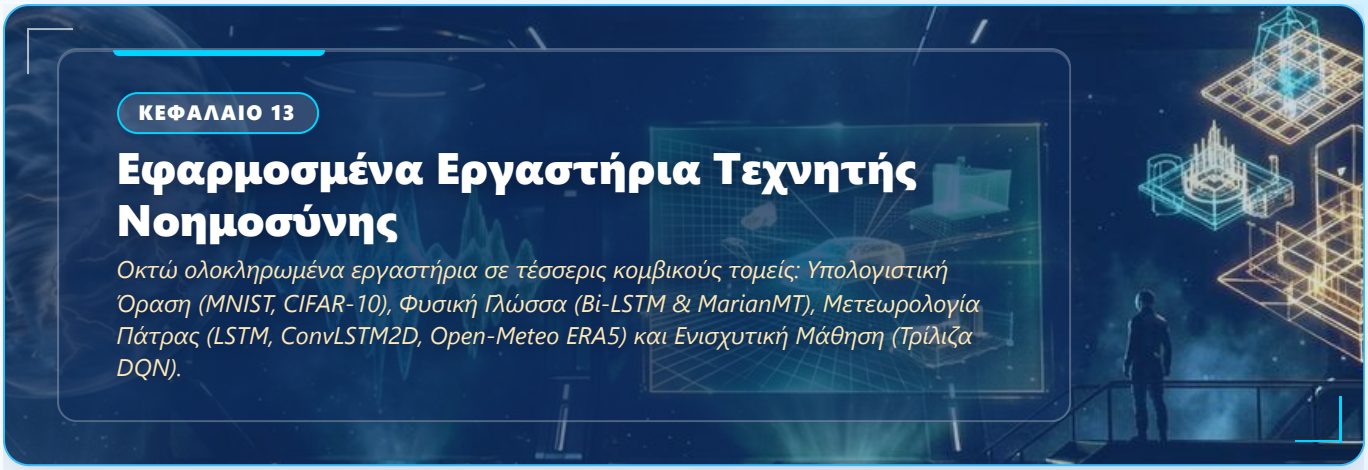
PYTHON 3.12+ `code/29_ai_agent_tool_calling_react.py` Εργαστήριο 29: Αυτόνομος πράκτορας ReAct & μητρώο εργαλείων MCP (JSON Tool Calling)

```
import json
import math


# === 1. Ορισμός & Μητρώο Ασφαλών Τοπικών Εργαλείων (Πρότυπο MCP) ===

def tool_gpu_vram_estimator(params_billions: float, quant_bits: float = 4.5) -> dict:
    """Υπολογίζει την εκτιμώμενη μνήμη GPU VRAM για τοπική εκτέλεση ανοιχτού LLM."""
    weights_gb = params_billions * (quant_bits / 8.0)
    kv_and_overhead = 0.9
    total = round(weights_gb + kv_and_overhead, 2)
    return {
        "model_params_B": params_billions,
        "quant_bits": quant_bits,
        "estimated_vram_gb": total,
        "fits_12gb_rtx3060": total <= 11.4,
    }
```

```python
def tool_calculator(expression: str) -> dict:
    """Ασφαλής απομονωμένος αποτιμητής μαθηματικών εκφράσεων (χωρίς ενσωματωμένες συναρτήσεις Python)."""
    allowed = {"sqrt": math.sqrt, "log2": math.log2, "pow": pow, "round": round}
    val = eval(expression, {"__builtins__": {}}, allowed)
    return {"expression": expression, "result": round(float(val), 4)}


def tool_search_syllabus(topic: str) -> dict:
    """Αναζητά το αντίστοιχο κεφάλαιο και εργαστήριο στην ύλη του σεμιναρίου Τμήματος Η.Μ.Μ.Υ."""
    index = {
        "lora": "Chapter XI (Fine-Tuning, LoRA & Alignment) -> Lab 27_lora_from_scratch_pytorch.py",
        "quantization": "Chapter X (Open-Source AI & Quantization) -> Lab
25_quantization_fp16_to_int4_math.py",
        "rag": "Chapter XII (RAG, Vector Databases & AI Agents) -> Lab
28_rag_vector_search_from_scratch.py",
    }
    key = next((k for k in index if k in topic.lower()), "quantization")
    return {"topic": topic, "courseware_reference": index[key]}


TOOL_REGISTRY = {
    "gpu_vram_estimator": tool_gpu_vram_estimator,
    "calculator": tool_calculator,
    "search_syllabus": tool_search_syllabus,
}


# === 2. Εκτελεστικός Βρόχος Αυτόνομου Πράκτορα ReAct ===

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/29_ai_agent_tool_calling_react.py του code.zip)
```

◈ Ο Χρυσός Κανόνας Επιλογής: Prompting vs. RAG vs. LoRA Fine-Tuning

1. Prompt Engineering: Ξεκινάμε πάντα εδώ για σαφείς οδηγίες και δομή εξόδου. **2. RAG & Εργαλεία MCP:** Όταν χρειαζόμαστε *ζωντανή, μεταβαλλόμενη γνώση, παραπομπές σε έγγραφα ή ακριβείς υπολογισμούς*. **3. LoRA / QLoRA Fine-Tuning:** Όταν θέλουμε να διδάξουμε στο μοντέλο ένα *νέο ύφος, εξειδικευμένη ορολογία ή δομημένη συμπεριφορά συλλογισμού* με 100% τοπικό έλεγχο.



ΚΕΦΑΛΑΙΟ 13

Εφαρμοσμένα Εργαστήρια Τεχνητής Νοημοσύνης

Οκτώ ολοκληρωμένα εργαστήρια σε τέσσερις κομβικούς τομείς: Υπολογιστική Όραση (MNIST, CIFAR-10), Φυσική Γλώσσα (Bi-LSTM & MarianMT), Μετεωρολογία Πάτρας (LSTM, ConvLSTM2D, Open-Meteo ERA5) και Ενισχυτική Μάθηση (Τρίλιζα DQN).

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- Εφαρμογή συνελικτικών δικτύων από την αναγνώριση χειρόγραφων ψηφίων **MNIST** έως έγχρωμες φυσικές εικόνες **CIFAR-10** με *Data Augmentation* και *BatchNormalization* (**Εργαστήρια 13 & 14**).
- Σύνθεση αμφίδρομης κατανόησης συμφραζομένων (**Bi-LSTM**) με προεκπαιδευμένη **Νευρωνική Μηχανική Μετάφραση (MarianMT)** Αγγλικά προς Ελληνικά (**Εργαστήριο 15**).
- Σχεδίαση **μετεωρολογικών αγωγών χρονοσειρών Πάτρας** χωρίς διαρροή δεδομένων σε **1D LSTM**, **5D ConvLSTM2D** και πραγματικά δεδομένα **Open-Meteo ERA5 API** (**Εργαστήρια 16, 17 & 18**).
- Εκπαίδευση και ζωντανή αλληλεπίδραση με αυτόνομο πράκτορα **Deep Q-Network (DQN)** στην Τρίλιζα μέσω Εξίσωσης Bellman, *Experience Replay* και πολιτικής *ε-greedy* (**Εργαστήρια 19 & 20**).

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Πώς Μετατρέπεται ο Πραγματικός Κόσμος σε Τανυστές; Οδηγός για τις 4 Εφαρμογές

Κάθε νευρωνικό δίκτυο καταλαβαίνει *αποκλειστικά αριθμούς*. Πριν εκτελέσετε τα οκτώ εργαστήρια αυτού του κεφαλαίου, δείτε πώς τέσσερις εντελώς διαφορετικοί κόσμοι κωδικοποιούνται σε καθαρούς αριθμητικούς τανυστές:

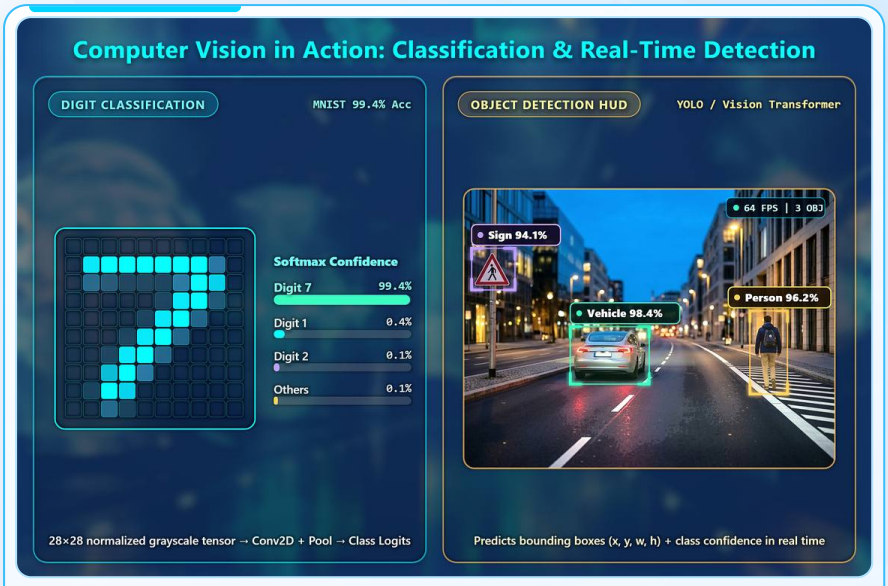
- **Όραση (Εργ. 13 & 14):** Κάθε ασπρόμαυρο ψηφίο MNIST γίνεται πίνακας φωτεινοτήτων `(28, 28, 1)` κανονικοποιημένος στο `[0, 1]` (διαίρεση με 255), ενώ κάθε έγχρωμη εικόνα CIFAR-10 γίνεται RGB τανυστής `(32, 32, 3)` (Κόκκινο, Πράσινο, Μπλε).
- **Φυσική Γλώσσα (Εργ. 15):** Κάθε λέξη κριτικής IMDb αντιστοιχίζεται σε έναν ακέραιο κωδικό λεξικού (token ID), δίνοντας 1D ακολουθία `(T,)` (π.χ. `[14, 22, 9, 43]`) που το επίπεδο `Embedding` μετατρέπει σε πυκνά διανύσματα νοήματος.
- **Μετεωρολογία Πάτρας (Εργ. 16 έως 18):** Ένα παράθυρο 14 ημερών με 4 μετρήσεις (θερμοκρασία, υγρασία, πίεση, άνεμος) γίνεται πίνακας `(14, 4)` για το 1D LSTM, ή αναδιαμορφώνεται σε χωροχρονικό πλέγμα `(14, 2, 2, 1)` (5D τανυστής με το batch) για το `ConvLSTM2D`.
- **Ενισχυτική Μάθηση Τρίλιζας (Εργ. 19 & 20):** Το ταμπλό 3×3 μετατρέπεται σε διάνυσμα 9 αριθμών με τιμές `[-1, 0, +1]` (*−1* = άνθρωπος X, *0* = κενό κελί, *+1* = πράκτορας AI O)!

13.1 Υπολογιστική Όραση: MNIST, CIFAR-10 & Ανίχνευση Αντικειμένων (YOLO / ViT)

Η υπολογιστική όραση μεταβαίνει κλιμακωτά από καθαρές μονοχρωματικές εικόνες χαμηλής διακύμανσης (**MNIST**: 70.000 χειρόγραφα ψηφία *28 × 28 × 1*) σε πολύπλοκες φυσικές έγχρωμες σκηνές (**CIFAR-10**: 60.000 εικόνες RGB *32 × 32 × 3* με μεταβαλλόμενο φωτισμό, υπόβαθρο και προοπτική). Ενώ ένα απλό CNN δύο βαθμίδων αρκεί για ακρίβεια **άνω του 99%** στο MNIST, η γενίκευση στο CIFAR-10 απαιτεί βαθύτερες στοίβες φίλτρων (32→64→128), ενσωματωμένη **Επαύξηση Δεδομένων (Data Augmentation)** (`RandomFlip, RandomTranslation`) που δημιουργεί γεωμετρικές παραλλαγές σε πραγματικό χρόνο μόνο κατά την εκπαίδευση, σταθεροποίηση **BatchNormalization** και **GlobalAveragePooling2D** που αντικαθιστά τα βαριά επίπεδα Flatten+Dense.

Στα σύγχρονα συστήματα αυτόνομης οδήγησης, ιατρικής απεικόνισης και ρομποτικής, αρχιτεκτονικές εντοπισμού ενός σταδίου όπως το **YOLO (You Only Look Once)** και τα **Vision Transformers (ViT)** (που διασπούν την εικόνα σε *patches* και εφαρμόζουν καθολική αυτοπροσοχή) προβλέπουν ταυτόχρονα συντεταγμένες πλαισίων *(x, y, w, h)*, εμπι-

στοσύνη και κλάσεις πολλαπλών αντικειμένων (με καταστολή μη μεγίστων *Non-Maximum Suppression*) σε ρυθμούς άνω των 60 καρέ ανά δευτερόλεπτο.



Σχήμα 13.1

Σύγχρονη υπολογιστική όραση: ταξινόμηση εικόνων και ανίχνευση αντικειμένων σε πραγματικό χρόνο (οχήματα, πεζοί, σήμανση).

ΕΡΓΑΣΤΗΡΙΟ 13 Αναγνώριση Χειρόγραφων Ψηφίων MNIST με Συνελικτικό Δίκτυο 2 Βαθμίδων

`code/13_mnist_digit_recognition.py`

Φορτώνει 60.000 εικόνες εκπαίδευσης και 10.000 αξιολόγησης, κλιμακώνει τις εντάσεις από `[0, 255]` στο `[0, 1]`, προσθέτει τον άξονα καναλιού `(28, 28, 1)` και εκπαιδεύει CNN (`Conv2D(32) → MaxPool → Conv2D(64) → MaxPool → Dropout(0.3) → Dense(128) → Softmax(10)`) με `sparse_categorical_crossentropy`, πετυχαίνοντας ακρίβεια >99%.

Προτεινόμενος Πειραματισμός: Απεικονίστε το πλέγμα προβλέψεων και εντοπίστε τα ελάχιστα ψηφία όπου σφάλλει το δίκτυο (συνήθως δυσανάγνωστα 4/9 ή 3/8): παρατηρήστε τις πιθανότητες Softmax της δεύτερης επιλογής του μοντέλου.

PYTHON 3.12+ `code/13_mnist_digit_recognition.py` CNN 2 βαθμίδων στο MNIST με Keras 3 (>99% ακρίβεια)

```python
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras import layers


def main() -> None:
    """Εκπαιδεύει CNN στο MNIST, αξιολογεί την ακρίβεια, αποθηκεύει το μοντέλο και σχεδιάζει προβλέψεις."""
    keras.utils.set_random_seed(42)

    # === 1. Φόρτωση & Κανονικοποίηση Εικόνων MNIST ===
    # 60.000 εικόνες εκπαίδευσης και 10.000 εικόνες αξιολόγησης (διαστάσεις 28x28)
    (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

    # Κλιμάκωση εντάσεων εικονοστοιχείων από [0, 255] στο [0, 1] και προσθήκη άξονα καναλιού (28, 28, 1)
    x_train = (x_train.astype(np.float32) / 255.0)[..., np.newaxis]
    x_test = (x_test.astype(np.float32) / 255.0)[..., np.newaxis]

    # === 2. Ορισμός & Μεταγλώττιση Αρχιτεκτονικής CNN ===
    model = keras.Sequential(
        [
            keras.Input(shape=(28, 28, 1), name="digit_image"),
            layers.Conv2D(32, kernel_size=3, activation="relu"),
            layers.MaxPooling2D(pool_size=2),
```

```
            layers.Conv2D(64, kernel_size=3, activation="relu"),
            layers.MaxPooling2D(pool_size=2),
            layers.Flatten(),
            layers.Dropout(0.3),
            layers.Dense(128, activation="relu"),
            layers.Dense(10, activation="softmax"),
        ],
        name="mnist_digit_cnn",
    )

    # Απώλεια sparse_categorical_crossentropy για ακέραιες ετικέτες κλάσεων (0 έως 9) χωρίς one-hot
    model.compile(
        optimizer=keras.optimizers.Adam(learning_rate=1e-3),
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/13_mnist_digit_recognition.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 14

Βαθύ Συνελικτικό Δίκτυο 3 Σταδίων για CIFAR-10 με Data Augmentation & BatchNormalization

`code/14_cifar10_cnn_vision.py`

Επεκτείνει την όραση σε 10 κατηγορίες έγχρωμων φυσικών εικόνων (Αεροπλάνο έως Φορτηγό) με ενσωματωμένη βαθμίδα `data_augmentation` (οριζόντια αναστροφή & μετατόπιση 8%), ζεύγη `Conv2D(padding='same')` + `BatchNormalization` ανά στάδιο, κλιμακούμενο `Dropout` και `GlobalAveragePooling2D`.

Προτεινόμενος Πειραματισμός: Απενεργοποιήστε προσωρινά τη βαθμίδα `data_augmentation` και συγκρίνετε το κενό μεταξύ `accuracy` εκπαίδευσης και `val_accuracy` επικύρωσης για να μετρήσετε ποσοτικά τη μείωση της υπερεκπαίδευσης.

PYTHON 3.12+ **code/14_cifar10_cnn_vision.py** Βαθύ CNN με Data Augmentation & BatchNormalization για CIFAR-10

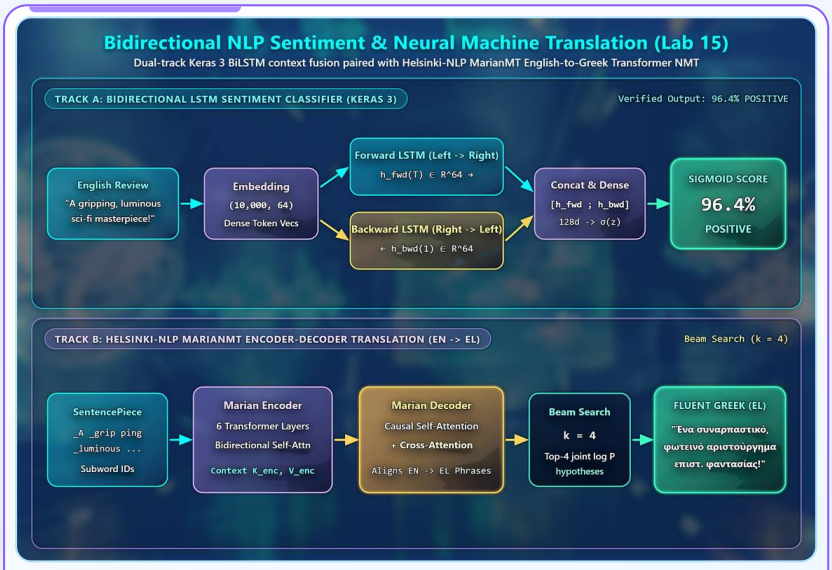
```
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import keras
from keras import layers

# Ελληνικές ετικέτες για τις 10 κατηγορίες αντικειμένων του συνόλου CIFAR-10
CLASS_NAMES_EL = [
    "Αεροπλάνο",
    "Αυτοκίνητο",
    "Πουλί",
    "Γάτα",
    "Ελάφι",
    "Σκύλος",
    "Βάτραχος",
    "Άλογο",
    "Πλοίο",
    "Φορτηγό",
]


def build_cifar10_cnn() -> keras.Model:
    """Κατασκευάζει βαθιά συνελικτική αρχιτεκτονική τριών σταδίων με Data Augmentation και
BatchNormalization."""
    # Ενσωματωμένοι γεωμετρικοί μετασχηματισμοί που εφαρμόζονται αυτόματα μόνο κατά την εκπαίδευση
    data_augmentation = keras.Sequential(
        [
            layers.RandomFlip("horizontal"),
            layers.RandomTranslation(0.08, 0.08),
        ],
        name="data_augmentation",
    )

    model = keras.Sequential(
        [
            keras.Input(shape=(32, 32, 3), name="rgb_image"),
            data_augmentation,
            # Στάδιο 1: εξαγωγή τοπικών ακμών και χρωματικών αντιθέσεων με σταθεροποίηση
```

```
BatchNormalization
            layers.Conv2D(32, 3, padding="same", activation="relu"),
            layers.BatchNormalization(),
            layers.Conv2D(32, 3, padding="same", activation="relu"),
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/14_cifar10_cnn_vision.py του code.zip)
```

13.2 Επεξεργασία Φυσικής Γλώσσας: Αμφίδρομο LSTM & Νευρωνική Μετάφραση (NMT)

Η κατανόηση φυσικής γλώσσας ξεκινά με τη μετατροπή διακριτών λέξεων σε πυκνά συνεχή διανύσματα μέσω ενός εκπαιδεύσιμου επιπέδου **Ενσωμάτωσης Λέξεων (Word Embedding)**, όπου σημασιολογικά συγγενείς έννοιες ομαδοποιούνται κοντά στον διανυσματικό χώρο. Για την ανάλυση συναισθήματος σε κινηματογραφικές κριτικές, ένα απλό μονόδρομο LSTM υστερεί όταν η άρνηση ή η ανατροπή εμφανίζεται στο τέλος της πρότασης. Ένα **Αμφίδρομο LSTM (Bidirectional LSTM)** διατρέχει την ακολουθία ταυτόχρονα από αριστερά προς δεξιά (h_{fwd}) και από δεξιά προς αριστερά (h_{bwd}), συγχωνεύοντας τις δύο καταστάσεις $[h_{fwd} ; h_{bwd}]$ ώστε κάθε λέξη να αξιολογείται υπό το πλήρες αμφίπλευρο πλαίσιο.

Για τη **Νευρωνική Μηχανική Μετάφραση (NMT)** από τα Αγγλικά στα Ελληνικά, οι σύγχρονες εφαρμογές βασίζονται σε αρχιτεκτονικές **Encoder-Decoder Transformer** όπως το ανοιχτό μοντέλο **Helsinki-NLP/opus-mt-en-el (MarianMT)**: ένας διαχωριστής υπολέξεων *SentencePiece* αναλύει σπάνιες λέξεις και κλίσεις, ο 6-επίπεδος Κωδικοποιητής παράγει αμφίδρομες αναπαραστάσεις (K_{enc}, V_{enc}) της αγγλικής πρότασης, ο Αποκωδικοποιητής ευθυγραμμίζει τις ελληνικές φράσεις μέσω **Διασταυρούμενης Προσοχής (Cross-Attention)** και η αναζήτηση δέσμης (**Beam Search, k = 4**) επιλέγει τη μετάφραση με τη μέγιστη συνδυαστική λογαριθμική πιθανότητα.



Σχήμα 13.2

Διπλός αγωγός Επεξεργασίας Φυσικής Γλώσσας (Εργαστήριο 15): Αμφίδρομο LSTM ανάλυσης συναισθήματος IMDb και Νευρωνική Μετάφραση MarianMT Αγγλικά προς Ελληνικά με Beam Search.

ΕΡΓΑΣΤΗΡΙΟ 15

Ανάλυση Συναισθήματος IMDb με Bi-LSTM & Νευρωνική Μετάφραση Αγγλικά προς Ελληνικά

`code/15_imdb_bilstm_and_translation.py`

Συνδυάζει δύο ολοκληρωμένους αγωγούς NLP: εκπαιδεύει επίπεδο `Embedding(10000, 64)` + `Bidirectional(LSTM(64))` πάνω στις 50.000 κριτικές του IMDb, αποκωδικοποιεί τα ακέραια token IDs πίσω σε αγγλικό κείμενο (αφαιρώντας τη μετατόπιση 3 των δεσμευμένων δεικτών) και καλεί το προεκπαιδευμένο **Helsinki-NLP/opus-mt-en-el** για αυτόματη απόδοση στα ελληνικά.

Προτεινόμενος Πειραματισμός: Επιλέξτε μία κριτική με ανάμεικτο ύφος ή σαρκασμό από το σύνολο αξιολόγησης και συγκρίνετε την πιθανότητα Sigmoid του Bi-LSTM με την ελληνική μετάφραση του MarianMT.

PYTHON 3.12+ `code/15_imdb_bilstm_and_translation.py` Αμφίδρομο LSTM & Νευρωνική Μετάφραση Αγγλικά προς Ελληνικά

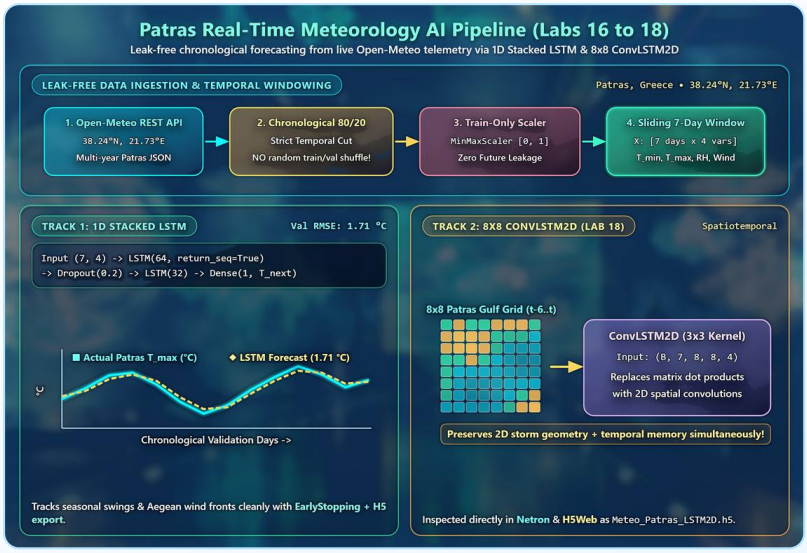
```python
import numpy as np
import keras
from keras import layers

try:
    from transformers import AutoModelForSeq2SeqLM, AutoTokenizer
except ImportError:
    AutoModelForSeq2SeqLM = None
    AutoTokenizer = None


def decode_review(encoded_tokens: np.ndarray, reverse_index: dict[int, str]) -> str:
    """Μετατρέπει ακολουθία ακέραιων δεικτών IMDb πίσω σε αναγνώσιμο αγγλικό κείμενο."""
    # Στο IMDb οι δείκτες 0, 1 και 2 δεσμεύονται για padding, έναρξη ακολουθίας και άγνωστες λέξεις
    words = [reverse_index.get(token - 3, "") for token in encoded_tokens if token >= 3]
    return " ".join(w for w in words if w)


def translate_en_to_el(text: str, max_chars: int = 450) -> str:
    """Μεταφράζει αγγλικό απόσπασμα στα ελληνικά με το προεκπαιδευμένο μοντέλο Helsinki-NLP/opus-mt-en-el."""
    if AutoTokenizer is None or AutoModelForSeq2SeqLM is None:
        return "(Εγκαταστήστε τις βιβλιοθήκες transformers, torch και sentencepiece για μετάφραση.)"

    model_id = "Helsinki-NLP/opus-mt-en-el"
    tokenizer = AutoTokenizer.from_pretrained(model_id)
    translator = AutoModelForSeq2SeqLM.from_pretrained(model_id)

    # Τεκμηριωμένη περικοπή και αυτοπαλίνδρομη παραγωγή μετάφρασης Seq2Seq
    snippet = text[:max_chars]
    inputs = tokenizer(snippet, return_tensors="pt", truncation=True, max_length=256)
    output_ids = translator.generate(**inputs, max_new_tokens=256)
    return tokenizer.decode(output_ids[0], skip_special_tokens=True)


def main() -> None:
    """Εκπαιδεύει Bi-LSTM στο IMDb, αξιολογεί την ακρίβεια και μεταφράζει τυχαία κριτική."""
    keras.utils.set_random_seed(42)

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/15_imdb_bilstm_and_translation.py του code.zip)
```

13.3 Μετεωρολογία Πάτρας: Πολυμεταβλητό LSTM, Χωροχρονικό ConvLSTM2D & Open-Meteo ERA5 API

Η μετεωρολογική πρόγνωση αποτελεί κατεξοχήν πρόβλημα **συζευγμένων πολυμεταβλητών χρονοσειρών**: η θερμοκρασία, η σχετική υγρασία, η ατμοσφαιρική πίεση και η ταχύτητα ανέμου αλληλεπιδρούν φυσικά υπό τον μεσογειακό εποχικό κύκλο και τη συνοπτική επιμονή των καιρικών μετώπων (μοντελοποιούμενη ως αυτοπαλίνδρομη διαδικασία

AR(1)). Η χρυσή αρχή των τριών μετεωρολογικών εργαστηρίων μας είναι η **μηδενική διαρροή μελλοντικής πληροφορίας**: διαχωρίζουμε αυστηρά χρονολογικά (π.χ. 85% παρελθόν εκπαίδευσης, 15% μελλοντική αξιολόγηση), προσαρμόζουμε το `MinMaxScaler` *μόνο* στο παρελθόν και σχηματίζουμε ολισθαίνοντα χρονικά παράθυρα 14 ημερών.

Στη συνέχεια προχωρούμε ένα βήμα πέρα από το 1D LSTM: αναδιαμορφώνουμε τις τέσσερις συζευγμένες μετεωρολογικές μεταβλητές κάθε ημέρας σε δισδιάστατο πλέγμα `2 × 2 × 1` (σχηματίζοντας 5D τανυστή `(B, 14, 2, 2, 1)`) ώστε ένα επίπεδο **ConvLSTM2D** να επεξεργάζεται ταυτόχρονα τις διασταυρούμενες συσχετίσεις των μεταβλητών και τη χρονική τους μνήμη. Τέλος, συνδέουμε τον αγωγό με πραγματικές δορυφορικές και επίγειες μετρήσεις επαναανάλυσης **ERA5** για την Πάτρα (**38.2466° Β, 21.7346° Α**) μέσω του ανοιχτού **Open-Meteo Archive REST API** χωρίς καμία απαίτηση κλειδιού API.



Σχήμα 13.3

Ολοκληρωμένος αγωγός μετεωρολογικής πρόγνωσης Πάτρας χωρίς διαρροή δεδομένων (Εργαστήρια 16 έως 18): ανάκτηση Open-Meteo API, αυστηρός χρονικός διαχωρισμός, 1D LSTM και χωροχρονικό ConvLSTM2D.

ΕΡΓΑΣΤΗΡΙΟ 16 **Πολυμεταβλητή Πρόγνωση Καιρού Πάτρας με LSTM & Κανονικοποίηση Χωρίς Διαρροή**

`code/16_patras_meteo_lstm.py`

Συνθέτει (ή φορτώνει) 25ετή ημερήσια κλιματική χρονοσειρά Πάτρας 4 συζευγμένων παραμέτρων (`temperature_c, humidity_pct, pressure_hpa, wind_kmh`), εφαρμόζει αυστηρά χρονικό split, προσαρμόζει `MinMaxScaler` μόνο στο Training Split, εκπαιδεύει πολυμεταβλητό **LSTM** σε ολισθαίνον παράθυρο 14 ημερών και αποκαθιστά τις φυσικές μονάδες με `inverse_transform` για υπολογισμό πραγματικού RMSE σε °C.

Προτεινόμενος Πειραματισμός: Συγκρίνετε το RMSE πρόβλεψης του LSTM με το *απλοϊκό μοντέλο επιμονής (persistence baseline)* που προβλέπει απλώς ότι η αυριανή θερμοκρασία ισούται με τη σημερινή: κάθε σοβαρό μοντέλο χρονοσειρών οφείλει να ξεπερνά αυτό το σημείο αναφοράς!

PYTHON 3.12+ `code/16_patras_meteo_lstm.py` Κανονικοποίηση χωρίς διαρροή & πολυμεταβλητό LSTM Πάτρας

```python
from pathlib import Path
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import keras
from keras import layers

# Τέσσερις μετεωρολογικές μεταβλητές εισόδου και πρόβλεψης
FEATURE_COLS = ["temperature_c", "humidity_pct", "pressure_hpa", "wind_kmh"]


def generate_patras_climate_csv(csv_path: Path, start: str = "2000-01-01", end: str = "2024-12-31") ->
pd.DataFrame:
    """Δημιουργεί ρεαλιστική ημερήσια κλιματική χρονοσειρά Πάτρας με εποχική περιοδικότητα και AR(1)."""
```

```
    dates = pd.date_range(start=start, end=end, freq="D")
    n = len(dates)
    day_of_year = dates.dayofyear.to_numpy(dtype=np.float64)
    rng = np.random.default_rng(seed=42)

    # Μεσογειακός εποχικός κύκλος: μέγιστη θερμοκρασία τέλη Ιουλίου και ελάχιστη τον Ιανουάριο
    seasonal = np.sin(2.0 * np.pi * (day_of_year - 110.0) / 365.25)

    # Αυτοπαλίνδρομη διαδικασία πρώτης τάξης AR(1) που προσομοιώνει την επιμονή καιρικών μετώπων
    weather_front = np.zeros(n, dtype=np.float64)
    for i in range(1, n):
        weather_front[i] = 0.72 * weather_front[i - 1] + rng.normal(0.0, 1.3)

    # Συσχετισμένες φυσικές μεταβλητές με ρεαλιστικά όρια αποκοπής (clip)
    temperature = 18.2 + 9.4 * seasonal + weather_front
    humidity = np.clip(67.0 - 13.5 * seasonal - 1.1 * weather_front + rng.normal(0.0, 4.5, n), 28.0, 96.0)
    pressure = np.clip(1015.5 + 3.2 * seasonal - 1.6 * weather_front + rng.normal(0.0, 2.5, n), 992.0,
1036.0)
    wind = np.clip(13.0 - 2.0 * seasonal + 0.9 * np.abs(weather_front) + rng.normal(0.0, 3.0, n), 1.5,
58.0)

    df = pd.DataFrame(
        {
            "date": dates,
            "temperature_c": np.round(temperature, 2),
            "humidity_pct": np.round(humidity, 2),
            "pressure_hpa": np.round(pressure, 2),
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/16_patras_meteo_lstm.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 17 **Χωροχρονική Πρόγνωση Καιρού Πάτρας με 5D ConvLSTM2D, BatchNorm & TensorBoard**

`code/17_patras_meteo_convlstm2d.py`

Αναδιαμορφώνει κάθε χρονικό παράθυρο 14 ημερών σε πενταδιάστατο τανυστή `(samples, 14, 2, 2, 1)` μέσω `make_convlstm_windows`, εκπαιδεύει `ConvLSTM2D(filters=32, kernel_size=(2, 2)) + BatchNormalization` με ταυτόχρονη καταγραφή **TensorBoard** και **EarlyStopping**, αποθηκεύει το `meteo_patras_convlstm2d.keras` και σχεδιάζει καμπύλη 90 ημερών σε °C.

Προτεινόμενος Πειραματισμός: Ανοίξτε το `meteo_patras_convlstm2d.keras` στο **Netron** για να παρατηρήσετε τον 5D τανυστή εισόδου και τις εσωτερικές 2D συνελικτικές πύλες του ConvLSTM2D.

PYTHON 3.12+ `code/17_patras_meteo_convlstm2d.py` Χωροχρονική αρχιτεκτονική 5D ConvLSTM2D Πάτρας

```
from datetime import datetime
from pathlib import Path
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
import keras
from keras import layers


# === 1. Σταθερές & Φόρτωση / Σύνθεση Μετεωρολογικών Δεδομένων ===

FEATURE_COLS = ["temperature_c", "humidity_pct", "pressure_hpa", "wind_kmh"]


def load_or_create_dataset(csv_path: Path) -> pd.DataFrame:
    """Φορτώνει το αρχείο patras_meteo.csv ή παράγει συνθετική χρονοσειρά 20 ετών."""
    if csv_path.exists():
        return pd.read_csv(csv_path, parse_dates=["date"])

    # Παραγωγή ημερήσιας ακολουθίας 20 ετών με ετήσια εποχικότητα και αυτοπαλίνδρομο μέτωπο AR(1)
    dates = pd.date_range(start="2005-01-01", end="2024-12-31", freq="D")
    n = len(dates)
    doy = dates.dayofyear.to_numpy(dtype=np.float64)
```

```
    rng = np.random.default_rng(seed=108)
    seasonal = np.sin(2.0 * np.pi * (doy - 110.0) / 365.25)

    # Μοντελοποίηση συνοπτικών καιρικών μεταβολών με συντελεστή επιμονής 0.75
    front = np.zeros(n, dtype=np.float64)
    for i in range(1, n):
        front[i] = 0.75 * front[i - 1] + rng.normal(0.0, 1.2)

    df = pd.DataFrame(
        {
            "date": dates,
            "temperature_c": np.round(18.2 + 9.4 * seasonal + front, 2),
            "humidity_pct": np.round(np.clip(67.0 - 13.5 * seasonal - front + rng.normal(0, 4.0, n), 30,
96), 2),
            "pressure_hpa": np.round(np.clip(1015.5 + 3.0 * seasonal - 1.5 * front + rng.normal(0, 2.2,
n), 993, 1035), 2),
            "wind_kmh": np.round(np.clip(13.0 - 1.8 * seasonal + np.abs(front) + rng.normal(0, 2.8, n),
1.5, 55), 2),
        }
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/17_patras_meteo_convlstm2d.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 18 **Αυτοματοποιημένη Ανάκτηση Πραγματικών Μετρήσεων ERA5 Πάτρας (Open-Meteo API)**

`code/18_patras_real_weather_api.py`

Χρησιμοποιώντας αποκλειστικά την τυπική βιβλιοθήκη της Python (`urllib, json, csv`), δομεί ερώτημα HTTP GET προς το δωρεάν επιστημονικό **Open-Meteo Archive API** για τις συντεταγμένες της Πάτρας (`38.2466, 21.7346`), καθαρίζει αυτόματα ημέρες με ελλιπείς τιμές (`None`) και παράγει το `patras_weather_real.csv` με το ίδιο ακριβώς σχήμα στηλών των Εργαστηρίων 16 και 17.

Προτεινόμενος Πειραματισμός: Μετονομάστε το παραγόμενο `patras_weather_real.csv` σε `patras_meteo.csv` και εκτελέστε εκ νέου τα Εργαστήρια 16 και 17 πάνω σε 100% πραγματικές ιστορικές μετεωρολογικές παρατηρήσεις της Πάτρας!

PYTHON 3.12+ **code/18_patras_real_weather_api.py** Ανάκτηση, καθαρισμός JSON & εξαγωγή CSV από Open-Meteo ERA5

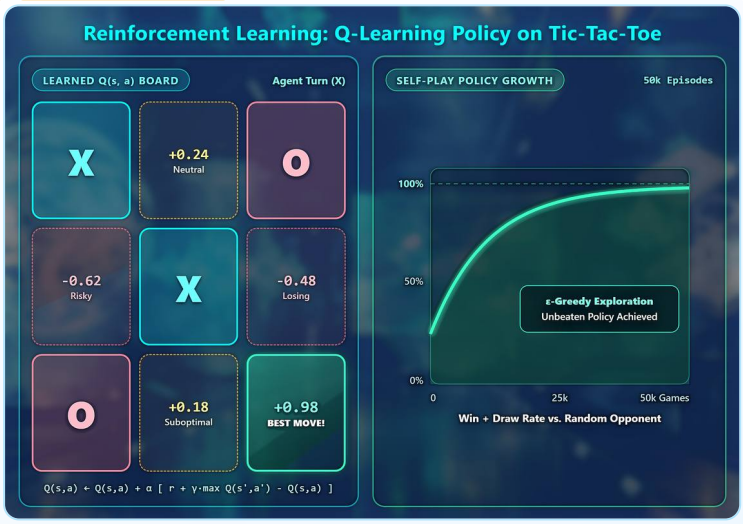
```
import csv
import json
from datetime import date, timedelta
from pathlib import Path
from urllib.error import URLError
from urllib.parse import urlencode
from urllib.request import urlopen


# === 1. Γεωγραφικές Συντεταγμένες Πάτρας & Τελικό Σημείο API ===

PATRAS_LAT = 38.2466
PATRAS_LON = 21.7346
API_URL = "https://archive-api.open-meteo.com/v1/archive"


# === 2. Ανάκτηση, Καθαρισμός & Εξαγωγή Ημερήσιων Μετρήσεων σε CSV ===

def fetch_patras_weather(start_date: str, end_date: str, out_csv: Path) -> int:
    """Ανακτά ημερήσιες μετρήσεις από το Open-Meteo Archive API και τις εγγράφει σε CSV."""
    params = {
        "latitude": PATRAS_LAT,
        "longitude": PATRAS_LON,
        "start_date": start_date,
        "end_date": end_date,
        "daily": ",".join(
            [
                "temperature_2m_mean",
                "relative_humidity_2m_mean",
                "pressure_msl_mean",
                "wind_speed_10m_max",
```

```
            ]
        ),
        "timezone": "Europe/Athens",
    }
    url = f"{API_URL}?{urlencode(params)}"

    # Εκτέλεση αιτήματος HTTP GET με χρονικό όριο 30 δευτερολέπτων και ανάλυση JSON
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/18_patras_real_weather_api.py του code.zip)
```

13.4 Ενισχυτική Μάθηση (Deep Q-Network, DQN): Εξίσωση Bellman, Experience Replay & Τρίλιζα

Στην **Ενισχυτική Μάθηση (Reinforcement Learning, RL)** δεν υπάρχει «δάσκαλος» που υποδεικνύει εκ των προτέρων τη σωστή ενέργεια. Ο πράκτορας αλληλεπιδρά με ένα περιβάλλον (εδώ το ταμπλό 9 θέσεων της Τρίλιζας), λαμβάνει σποραδικές ανταμοιβές (+1 νίκη, +0.2 ισοπαλία, −1 ήττα) και μαθαίνει τη βέλτιστη **συνάρτηση αξίας κατάστασης-ενέργειας Q*(s, a)**, δηλαδή τη συνολική αναμενόμενη προεξοφλημένη ανταμοιβή αν εκτελέσει την κίνηση *a* στην κατάσταση *s* και συνεχίσει βέλτιστα στη συνέχεια.

Όταν προσεγγίζουμε τη συνάρτηση $Q_\theta(s, a)$ με ένα βαθύ νευρωνικό δίκτυο (**Deep Q-Network, DQN**, Mnih et al., 2015), η απλή επιγραμμική εκπαίδευση αποσταθεροποιείται επειδή διαδοχικές κινήσεις είναι ισχυρά συσχετισμένες και ο στόχος Bellman μετακινείται συνεχώς. Η σύγχρονη αρχιτεκτονική DQN επιτυγχάνει σταθερή σύγκλιση χάρη σε **τρεις μηχανισμούς**: **(1) Μνήμη Επανάληψης Εμπειριών (Experience Replay Buffer)** που αποθηκεύει μεταβάσεις *(s, a, r, s', done)* σε κυκλική ουρά και δειγματοληπτεί τυχαίες δέσμες αποσυσχετίζοντας τον χρόνο, **(2) Δίκτυο Στόχου (Target Network Q_{θ^-})** με περιοδικά παγωμένα βάρη που σταθεροποιεί την τιμή-στόχο Bellman αποτρέποντας την ταλάντωση του «κινούμενου στόχου», και **(3) Πολιτική Εξερεύνησης ε-greedy** που μειώνει εκθετικά την πιθανότητα τυχαίας εξερεύνησης από $\varepsilon = 1.0$ προς $\varepsilon_{min} = 0.08$.



Σχήμα 13.4

Κύκλος Ενισχυτικής Μάθησης Deep Q-Learning στην Τρίλιζα: Κατάσταση ταμπλό, Δίκτυο Q-Values, Εξίσωση Bellman και Experience Replay.

$$Q_{target}(s, a) = r + \gamma \cdot \max_{a'} Q_{\theta^-}(s', a') \quad | \quad L(\theta) = E_{(s,a,r,s')\sim Replay}[\,(Q_{target}(s, a) - Q_\theta(s, a))^2\,]$$

Σφάλμα Χρονικής Διαφοράς (Temporal-Difference Error): ο συντελεστής προεξόφλησης $\gamma \in (0, 1)$ (π.χ. $\gamma = 0.92$) σταθμίζει την αξία των μελλοντικών κινήσεων έναντι της άμεσης ανταμοιβής r.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Μία Κίνηση Ενημέρωσης Q-Value κατά Bellman στο Χέρι στην Τρίλιζα

Έστω ότι ο πράκτορας στην κατάσταση ταμπλό s εκτιμά αρχικά την αξία μιας κίνησης a (π.χ. κατάληψη επάνω δεξιάς γωνίας) ως $Q_{old}(s, a) = \mathbf{0.400}$. Εκτελεί την κίνηση a, λαμβάνει άμεση ανταμοιβή $r = \mathbf{+1.0}$ και οδηγείται στη νέα κατάσταση s', όπου η καλύτερη μελλοντική απάντηση εκτιμάται σε $\max_{a'} Q(s', a') = \mathbf{0.800}$. Με ρυθμό μάθησης $\alpha = 0.1$ και προεξόφληση $\gamma = 0.9$:

- **Βήμα 1 (Στόχος Bellman):** $Q_{target} = r + \gamma \cdot \max_{a'} Q(s', a') = 1.0 + 0.9 \times 0.8 = 1.0 + 0.72 = \mathbf{1.720}$.
- **Βήμα 2 (Σφάλμα Χρονικής Διαφοράς / TD-Error):** $\delta = Q_{target} - Q_{old}(s, a) = 1.720 - 0.400 = \mathbf{+1.320}$ (θετική έκπληξη: η κίνηση αποδείχθηκε πολύ ισχυρότερη από την αρχική εκτίμηση!).
- **Βήμα 3 (Ενημέρωση Q-Value):** $Q_{new}(s, a) = 0.400 + 0.1 \times (1.0 + 0.9 \times 0.8 - 0.400) = 0.400 + 0.132 = \mathbf{0.532}$!

Σύνδεση με το DQN (Εργ. 19 & 20): Αντί να αποθηκεύουμε τεράστιο πίνακα για κάθε πιθανό ταμπλό, το νευρωνικό δίκτυο `9 → 64 → 64 → 9` προσαρμόζει τα βάρη του μέσω Backpropagation ώστε η πρόβλεψή του να πλησιάζει το Q_{target} (ενώ σε τερματική νίκη/ήττα δεν υπάρχουν επόμενες κινήσεις, άρα απλώς $Q_{target} = r$).

ΕΡΓΑΣΤΗΡΙΟ 19 **Εκπαίδευση Αυτόνομου Πράκτορα Τρίλιζας με Deep Q-Learning & Experience Replay**

`code/19_triliza_dqn_train.py`

Εκπαιδεύει MLP (`9 → 64 → 64 → 9` με γραμμική έξοδο Q-τιμών και απώλεια MSE) επί 800 παρτίδες απέναντι σε ημιέξυπνο αντίπαλο, εναλλάσσοντας ποιος παίζει πρώτος (επίθεση & άμυνα). Διατηρεί `deque(maxlen=4000)` για Experience Replay, εφαρμόζει εκθετική απόσβεση `epsilon_decay=0.995` και αποθηκεύει το εκπαιδευμένο μοντέλο στο `triliza_dqn.keras`.

Προτεινόμενος Πειραματισμός: Αυξήστε τα επεισόδια εκπαίδευσης από `800` σε `2500` και παρατηρήστε την άνοδο του ποσοστού νικών και ισοπαλιών καθώς το ε προσεγγίζει το 0.08 και το δίκτυο κατακτά τις διαγώνιες «διχάλες» (forks).

PYTHON 3.12+ `code/19_triliza_dqn_train.py` Πυρήνας εκπαίδευσης Deep Q-Learning (Bellman + Experience Replay)

```python
from collections import deque
from pathlib import Path
import random
import time
import numpy as np
import keras
from keras import layers


# === 1. Κανόνες Περιβάλλοντος Τρίλιζας ===

WIN_LINES = (
    (0, 1, 2), (3, 4, 5), (6, 7, 8),
    (0, 3, 6), (1, 4, 7), (2, 5, 8),
    (0, 4, 8), (2, 4, 6),
)


def check_winner(board: np.ndarray) -> int:
    """Επιστρέφει +1 αν κέρδισε ο πράκτορας, -1 αν κέρδισε ο αντίπαλος, αλλιώς 0."""
    for a, b, c in WIN_LINES:
        s = int(board[a] + board[b] + board[c])
        if s == 3:
            return 1
        if s == -3:
            return -1
    return 0


def valid_actions(board: np.ndarray) -> list[int]:
    """Επιστρέφει τους δείκτες (0 έως 8) όλων των κενών θέσεων στο ταμπλό."""
    return [i for i in range(9) if board[i] == 0]
```

```
# === 2. Αρχιτεκτονική Δικτύου DQN & Πολιτικές Επιλογής Κίνησης ===

def build_dqn_model() -> keras.Model:
    """Κατασκευάζει δίκτυο MLP που δέχεται τις 9 θέσεις του ταμπλό και εκτιμά 9 τιμές Q."""
    model = keras.Sequential(
        [
            keras.Input(shape=(9,), name="board_state"),
            layers.Dense(64, activation="relu"),
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/19_triliza_dqn_train.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 20 **Διαδραστική Τρίλιζα Tkinter (Άνθρωπος vs AI) & Ζωντανή Επιγραμμική Μάθηση**

`code/20_triliza_play_gui.py`

Φορτώνει το εκπαιδευμένο `triliza_dqn.keras` σε διαδραστικό πλέγμα 3×3 **Tkinter** όπου ο άνθρωπος (X) αντιμετωπίζει τον πράκτορα DQN (O). Στη λήξη κάθε αγώνα, η μέθοδος `_learn_from_game` διαδίδει αναδρομικά την τελική ανταμοιβή με προεξόφληση $\gamma = 0.90$ σε όλες τις κινήσεις του AI (`train_on_batch`) και αποθηκεύει αμέσως τα βελτιωμένα βάρη στον δίσκο!

Προτεινόμενος Πειραματισμός: Επιχειρήστε να νικήσετε επανειλημμένα τον πράκτορα με την ίδια ακριβώς παγίδα γωνιών: θα παρατηρήσετε ότι έπειτα από λίγες παρτίδες η επιγραμμική ενημέρωση Bellman προσαρμόζει τα Q-values ώστε να μπλοκάρει τη στρατηγική σας.

PYTHON 3.12+ `code/20_triliza_play_gui.py` Γραφικό περιβάλλον Tkinter & επιγραμμική ενημέρωση DQN

```
from pathlib import Path
import tkinter as tk
from tkinter import messagebox
import numpy as np
import keras
from keras import layers


# === 1. Κανόνες Παιχνιδιού & Φόρτωση / Αρχικοποίηση Δικτύου DQN ===

WIN_LINES = (
    (0, 1, 2), (3, 4, 5), (6, 7, 8),
    (0, 3, 6), (1, 4, 7), (2, 5, 8),
    (0, 4, 8), (2, 4, 6),
)


def check_winner(board: np.ndarray) -> int:
    """Επιστρέφει +1 αν κέρδισε ο πράκτορας AI, -1 αν κέρδισε ο άνθρωπος, αλλιώς 0."""
    for a, b, c in WIN_LINES:
        s = int(board[a] + board[b] + board[c])
        if s == 3:
            return 1
        if s == -3:
            return -1
    return 0


def load_or_init_model(model_path: Path) -> keras.Model:
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/20_triliza_play_gui.py του code.zip)
```

Κανόνες Επαναληψιμότητας στα Εφαρμοσμένα Εργαστήρια

Σε κάθε εφαρμοσμένο πείραμα τηρούμε τρεις αδιαπραγμάτευτες αρχές: **(1)** σταθερούς σπόρους τυχαιότητας (`seed=42`) για αναπαραγώγιμα αποτελέσματα, **(2)** αυστηρή αποφυγή διαρροής δεδομένων (ο κανονικοποιητής εκπαιδεύεται μόνο στο Training Split), και **(3)** αποθήκευση εκπαιδευμένων μοντέλων σε μορφή `.keras` ώστε η αξιολόγηση και η ζωντανή επίδειξη να είναι άμεσες.

ΚΕΦΑΛΑΙΟ 14

Ακρόαση του Σύμπαντος: Εκπαίδευση LLMs & Βαθιά Μάθηση στη Ραδιοαστρονομία SETI

Από τη ραδιοφυσική του Κοσμικού «Πηγαδιού του Νερού» και το πιλοτικό cluster Spark/VGG16 έως το νέο πείραμα SETI σε 1 GPU RTX 3060 12 GB (GUI), τα AstroSage/AION-1 και 4 ολοκληρωμένα εργαστήρια Python (Labs 30, 31, 32 & 33).

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Ραδιοφυσική, Κοσμικό «Πηγάδι του Νερού» & Δυναμικά Φασματογραφήματα:** Κατανόηση της αλυσίδας λήψης (Κεραία, Cryo LNA, ADC, PFB/FFT), της εξίσωσης ραδιομέτρου, της διασποράς ψυχρού πλάσματος (ν^{-2}), του ήσυχου παραθύρου 1.420 έως 1.662 GHz (H I / OH), του ρυθμού Doppler και των 6 κανονικών μορφολογιών σημάτων.
- **Νέο Πείραμα SETI σε 1 GPU RTX 3060 12 GB & Κατανεμημένη Βαθιά Μάθηση:** Μετεξέλιξη του πιλοτικού cluster 5 H/Y Spark/VGG16 (~88%) σε υβριδικό δίκτυο `SpectralResViTNet` (AMP BF16 + TF32), PyTorch DDP, αυστηρό χωρικό έλεγχο **ABACAD** και διδάγματα του *Breakthrough Listen* (`blc1`, β-CVAE).
- **Πολυτροπικά Astro-LLMs & Spectral ViT:** Σύνδεση φασματικών τεμαχίων (*patches*), αυτοεπιβλεπόμενων ανιχνευτών ανωμαλιών (MAE/SVD) και εξειδικευμένων μοντέλων (AstroLLaMA, AstroSage, AION-1) με πράκτορες IAA MCP.
- **Τέσσερα Πλήρη Ερευνητικά Εργαστήρια σε Python (Labs 30, 31, 32 & 33):** Εκτέλεση και πειραματική επαλήθευση από το ακατέργαστο ραδιοσήμα έως τη νέα γραφική εφαρμογή αστεροσκοπείου **RTX 3060 12 GB (Εργαστήριο 33)** και την τελική επιστημονική ετυμηγορία.

14.1 Ραδιοφυσική, Κοσμικό «Πηγάδι του Νερού» & η Πρόκληση των Δεδομένων SETI

Η σύγχρονη ραδιοαστρονομία και η επιστημονική αναζήτηση τεχνοϋπογραφών (**SETI, Search for Extraterrestrial Intelligence**) αποτελούν ίσως το πιο απαιτητικό πεδίο ανάλυσης σημάτων στον πλανήτη. Σύγχρονες συστοιχίες ραδιοτηλεσκοπίων όπως τα **Green Bank (GBT)**, **MeerKAT**, **FAST**, **LOFAR**, **ALMA** και το επερχόμενο **Square Kilometre Array (SKA)** καταγράφουν εκατοντάδες gigabits ανά δευτερόλεπτο φασματογραφικών δεδομένων (*dynamic filterbank spectrograms*) σε δισεκατομμύρια στενά κανάλια συχνότητας. Μέσα σε αυτόν τον ωκεανό θορύβου, ένα πραγματικό κοσμικό σήμα (είτε φυσικός **Πάλσαρ**, είτε **Ταχεία Έκρηξη Ραδιοκυμάτων, FRB**, είτε μια στενής ζώνης τεχνοϋπογραφή με ολίσθηση Doppler) κρύβεται πίσω από εκατομμύρια ανθρωπογενείς παρεμβολές ραδιοσυχνοτήτων (**RFI** από δορυφόρους, ραντάρ, GPS και κινητή τηλεφωνία).

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Πώς «Ακούει» ένα Ραδιοτηλεσκόπιο; Από το Αόρατο Κύμα στο Φασματόγραμμα Καταρράκτη

Φανταστείτε ένα γιγαντιαίο παραβολικό πιάτο διαμέτρου δεκάδων μέτρων ως ένα εξαιρετικά κατευθυντικό **κοσμικό μικρόφωνο**: αντί για ηχητικά κύματα, συλλέγει ασύλληπτα ασθενή ηλεκτρομαγνητικά κύματα που ταξίδεψαν εκατομμύρια ή δισεκατομμύρια έτη φωτός. Η πυκνότητα ροής τους μετριέται σε **Jansky** (`1 Jy = 10⁻²⁶ W m⁻² Hz⁻¹`), ένα μέγεθος δισεκατομμύρια δισεκατομμυρίων φορές πιο αδύναμο από το σήμα ενός κινητού τηλεφώνου!

Στην εστία της κεραίας, το αμυδρό ραδιοκύμα επάγει μια απειροελάχιστη ηλεκτρική τάση, η οποία ενισχύεται σε κρυογενικό περιβάλλον (~15 K), ψηφιοποιείται δισεκατομμύρια φορές το δευτερόλεπτο από έναν **ADC** και διαχωρίζεται σε εκατομμύρια λεπτά κανάλια συχνότητας μέσω **Πολυφασικής Τράπεζας Φίλτρων (PFB)** και **Γρήγορου Μετασχηματισμού Fourier (FFT)**. Στοιβάζοντας αυτά τα στιγμιαία φάσματα χρονικά το ένα πάνω στο άλλο, σχηματίζουμε μια δισδιάστατη εικόνα **Φασματογράμματος Καταρράκτη (Waterfall Spectrogram)** (Συχνότητα στον οριζόντιο άξονα, Χρόνος στον κατακόρυφο, Ισχύς ως φωτεινότητα). Έτσι, η ραδιοαστρονομία μετατρέπεται σε πρόβλημα υπολογιστικής όρασης: κάθε φυσικό φαινόμενο χαράζει ένα αλάνθαστο *γεωμετρικό αποτύπωμα* πάνω στην εικόνα!



Σχήμα 14.1

Συστοιχία παραβολικών ραδιοτηλεσκοπίων στραμμένη προς το κέντρο του Γαλαξία μέσα από το Κοσμικό «Πηγάδι του Νερού» (1.420 έως 1.662 GHz): αόρατα ραδιοκύματα αστροφυσικών πηγών και υποψήφιων τεχνοϋπογραφών μετατρέπονται σε ψηφιακά φασματογραφήματα υψηλής ανάλυσης.

Η Αλυσίδα Λήψης Ραδιοτηλεσκοπίου & οι Τρεις Θεμελιώδεις Εξισώσεις

Κεντρικό ερευνητικό ερώτημα (όπως τεκμηριώνεται και στην ερευνητική πρόταση του συγγραφέα στο Πανεπιστήμιο Πελοποννήσου) είναι πώς η **Βαθιά Μάθηση** και τα **Μεγάλα Γλωσσικά Μοντέλα (LLMs)** μετασχηματίζουν την αναγνώριση αστρονομικών σημάτων σε πραγματικό χρόνο, διακρίνοντας αξιόπιστα τα αυθεντικά αστροφυσικά και δυνητικά εξωγήινα σήματα από τις γήινες παρεμβολές. Σε επίπεδο μηχανικής συστημάτων, κάθε παρατήρηση ακολουθεί μια αυστηρή αλυσίδα επεξεργασίας σήματος: **Παραβολική Κεραία (Parabolic Dish) → Κρυογενικός Ενισχυτής Χαμηλού Θορύβου (Cryogenic LNA) → Μετατροπέας Αναλογικού σε Ψηφιακό (ADC) → Πολυφασική Τράπεζα Φίλτρων & FFT (PFB / FFT) → Αρχεία Ολικής Έντασης Stokes I Filterbank (`.fil / .h5`)**. Η μαθηματική θεμελίωση της προσομοίωσης ραδιοδεδομένων (κατά **Luo, Hobbs et al., CSIRO `simulateSearch`, MNRAS 2022**) συνοψίζεται σε τρεις θεμελιώδεις εξισώσεις ραδιοφυσικής:

$$S_{rms} = (2 \cdot k_B \cdot T_{sys}) / (A_e \cdot \sqrt{(n_p \cdot \Delta t \cdot \Delta \nu)}) \quad | \quad \Delta t_{DM} = 4.148808 \times 10^3 \cdot DM \cdot (\nu_{lo}^{-2} - \nu_{hi}^{-2}) \quad | \quad df/dt = (f_0 / c) \cdot a_{rel}$$

Εξίσωση Ραδιομέτρου (ευαισθησία δέκτη σε Jansky, με k_B σταθερά Boltzmann, T_{sys} θερμοκρασία συστήματος, A_e ενεργό εμβαδόν, n_p πολώσεις), Καθυστέρηση Διασποράς Ψυχρού Πλάσματος (s, DM σε pc cm^{-3}, ν σε MHz) και Ρυθμός Ολίσθησης Doppler (Hz/s).

Η πρώτη εξίσωση καθορίζει το θεμελιώδες όριο ευαισθησίας κάθε ραδιοτηλεσκοπίου: για να μειώσουμε τον θερμικό θόρυβο `$S_{rms}$` (και να αυξήσουμε τον λόγο σήματος προς θόρυβο `SNR = S / $S_{rms}$`), απαιτούνται μεγαλύτερη συλλεκτική επιφάνεια `$A_e$`, χαμηλότερη κρυογενική θερμοκρασία συστήματος `$T_{sys}$`, διπλή πόλωση (`$n_p = 2$`) και μεγαλύτερος χρόνος ολοκλήρωσης `Δt`. Η δεύτερη εξίσωση εξηγεί γιατί ένα εξωγαλαξιακό FRB δεν εμφανίζεται ως κατακόρυφη γραμμή στο φασματόγραμμα αλλά ως *παραβολική καμπύλη*: καθώς το ηλεκτρομαγνητικό κύμα διασχίζει τα ελεύθερα ηλεκτρόνια του μεσοαστρικού και διαγαλαξιακού ψυχρού πλάσματος, η ταχύτητα ομάδας των χαμηλότερων συχνοτήτων επιβραδύνεται αναλογικά του ν^{-2}. Με τη διαδικασία της **Ασύμφωνης Αποδιασποράς (Incoherent Dedispersion)**, ο αλγόριθμος μετατοπίζει χρονικά προς τα πίσω κάθε κανάλι συχνότητας για ένα πλέγμα δοκιμαστικών τιμών DM_{trial}: όταν προσεγγιστεί το αληθινό μέτρο διασποράς, η καμπύλη ευθυγραμμίζεται και καταρρέει σε έναν οξύ παλμό υψηλού λόγου σήματος προς θόρυβο ($S/N > 7\sigma$), αποκαλύπτοντας ταυτόχρονα την κοσμολογική απόσταση της πηγής!

Το Κοσμικό «Πηγάδι του Νερού» (Cosmic Water Hole: 1.420 έως 1.662 GHz)

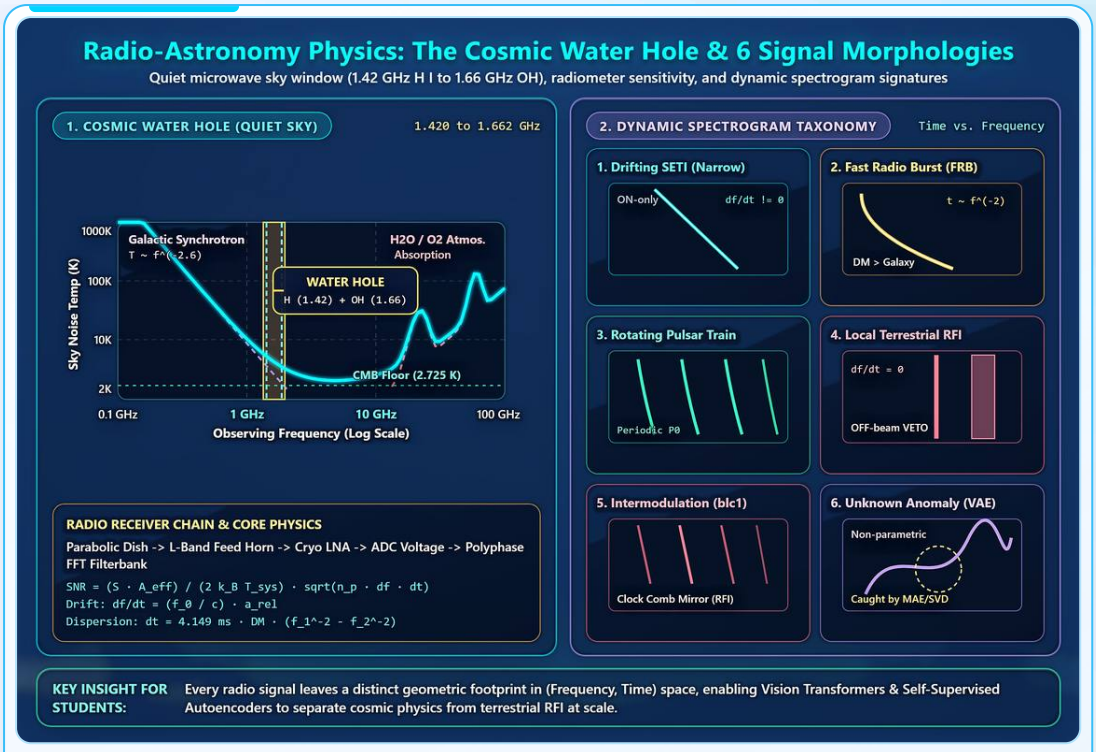
Μια θεμελιώδης ερώτηση στη ραδιοαστρονομία SETI είναι *σε ποια ζώνη του ηλεκτρομαγνητικού φάσματος αξίζει να ακούμε το Σύμπαν*. Η φυσική του ίδιου του Γαλαξία και της γήινης ατμόσφαιρας δίνει μια εντυπωσιακά σαφή απάντηση (όπως διατυπώθηκε ιστορικά από τον Bernard Oliver στο *Project Cyclops*): κάτω από το **1 GHz** κυριαρχεί η **γαλαξιακή ακτινοβολία συγχρότρου** από σχετικιστικά ηλεκτρόνια στο μαγνητικό πεδίο του Γαλαξία, ανεβάζοντας απότομα τη θερμοκρασία θορύβου του ουρανού σύμφωνα με τον νόμο $T_{sky} \propto f^{-2.6}$ (φτάνοντας έως τους 1000 K στα 0.1 GHz). Αντίστροφα, πάνω από τα **10 GHz** αυξάνεται απότομα η **ατμοσφαιρική απορρόφηση της Γης** από μόρια υδρατμών (H_2O κοντά στα 22.2 GHz) και μοριακού οξυγόνου (O_2 κοντά στα 60 GHz), σε συνδυασμό με τον κβαντικό θόρυβο φωτονίων.

Ανάμεσα στο 1 GHz και τα 10 GHz σχηματίζεται έτσι ένα εξαιρετικά ήσυχο φυσικό μικροκυματικό παράθυρο, όπου ο θόρυβος του ουρανού πλησιάζει το κοσμολογικό κατώφλι της **Κοσμικής Ακτινοβολίας Υποβάθρου (CMB, 2.725 K)**. Ακριβώς στο κέντρο αυτής της ήσυχης κοιλάδας δεσπόζουν δύο παγκόσμιες φασματικές γραμμές εκπομπής: η **υπέρλεπτη γραμμή ουδέτερου ατομικού υδρογόνου H I στα 1.420405 GHz (μήκος κύματος 21 cm)** και οι **γραμμές της ρίζας υδροξυλίου OH στα 1.612 έως 1.720 GHz (κύριο όριο 1.662 έως 1.667 GHz)**. Επειδή το υδρογόνο (**H**) και το υδροξύλιο (**OH**) συνθέτουν το μόριο του νερού (**H_2O**), η ζώνη **1.420 έως 1.662 GHz** ονομάστηκε **Κοσμικό «Πηγάδι του Νερού» (Cosmic Water Hole)**: ένα φυσικό, παγκόσμιο σημείο συνάντησης για κάθε τεχνολογικό πολιτισμό που γνωρίζει τη φυσική και τη χημεία της ζωής.

Οι Έξι Κανονικές Μορφολογίες Δυναμικών Φασματογραμμάτων

Κάθε σύγχρονος ταξινομητής βαθιάς μάθησης στη ραδιοαστρονομία καλείται να διακρίνει με ακρίβεια έξι θεμελιώδεις φασματοχρονικές γεωμετρίες, οι οποίες απεικονίζονται συγκεντρωτικά στο **Σχήμα 14.2**:

Μορφολογία Σήματος	Γεωμετρικό Αποτύπωμα στο Φασματόγραμμα (f, t)	Κρίσιμο Φυσικό Κριτήριο Διαχωρισμού
1. Θερμικός Θόρυβος Ραδιομέτρου	Ομοιόμορφος γκαουσιανός λευκός θόρυβος με ήπια διακύμανση ζώνης (bandpass) και $1/f^2$	Μηδενική φασματική συνοχή, στατιστική κατανομή καθορισμένη από την εξίσωση S_{rms}
2. Τεχνοϋπογραφή SETI (Doppler)	Εξαιρετικά στενή (λίγα Hz) κεκλιμένη γραμμή με συνεχή κλίση $df/dt \neq 0$	Εμφάνιση *αποκλειστικά* στα ON καρέ (A1, A2, A3) και πλήρης απουσία στα OFF (B, C, D)
3. Εξωγαλαξιακό FRB (ν^{-2})	Ευρυζωνικός παροδικός παλμός χιλιοστών δευτερολέπτου με έντονη παραβολική καμπύλη	Μέτρο διασποράς $DM > DM_{MW}$ (υπερβαίνει τη γαλαξιακή συμβολή)
4. Περιοδικός Πάλσαρ	Περιοδική ακολουθία ελαφρώς κεκλιμένων παλμών με σταθερή περίοδο P_0	Σταθερή φάση περιστροφής αστέρα νετρονίων και χαρακτηριστική ουρά σκέδασης
5. Επίγεια RFI & Κτένες `blc1`	Κατακόρυφες γραμμές ($df/dt = 0$), παλμικές ζώνες ραντάρ/GPS ή παράλληλες αρμονικές κτένες	Διαρροή από πλευρικούς λοβούς της κεραίας στα OFF καρέ → άμεση απόρριψη **ABACAD VETO**
6. Άγνωστη Κοσμική Ανωμαλία	Μη παραμετρική ημιτονοειδής, κυματοειδής ή σύνθετη φασματοχρονική διαμόρφωση	Εντοπισμός εκτός κλάσεων μέσω υψηλού σφάλματος ανακατασκευής (Autoencoder / MAE / SVD)



Σχήμα 14.2

Φυσική Ραδιοαστρονομίας: το ήσυχο Κοσμικό «Πηγάδι του Νερού» (1.420 έως 1.662 GHz ανάμεσα στον γαλαξιακό θόρυβο συγχρότρου και την ατμοσφαιρική απορρόφηση H2O/O2), η αλυσίδα λήψης ραδιομέτρου και η ταξινόμηση των 6 κανονικών γεωμετρικών μορφολογιών σε δυναμικά φασματογραφήματα.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

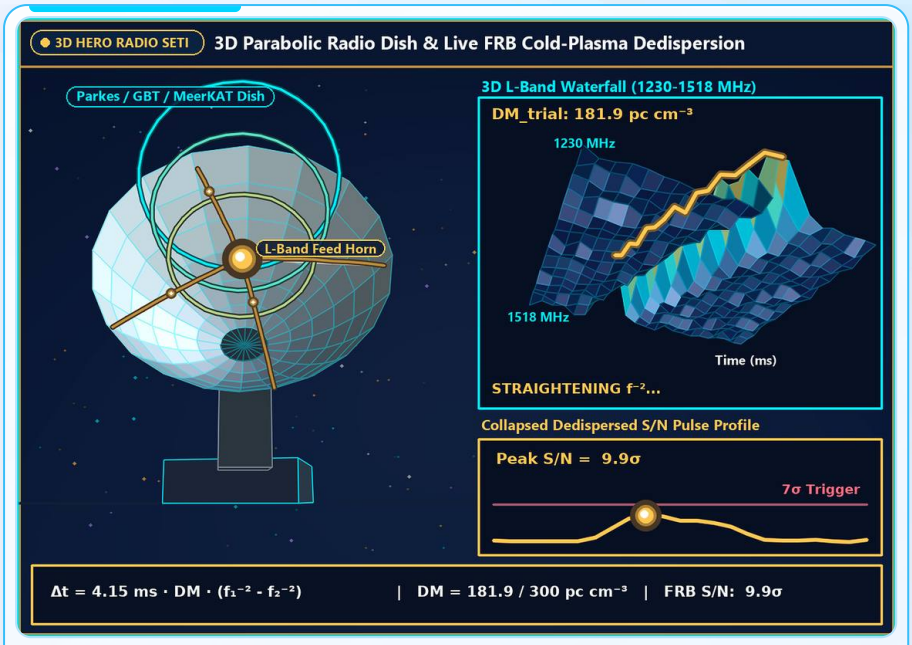
Υπολογισμός Καθυστέρησης Διασποράς FRB & Ρυθμού Ολίσθησης Doppler στο Χέρι

Μέρος Α: Εξωγαλαξιακή Καθυστέρηση Διασποράς FRB στη ζώνη L-Band. Έστω ότι ένα ραδιοτηλεσκόπιο παρατηρεί από $v_{lo} = 1230$ MHz έως $v_{hi} = 1518$ MHz ένα FRB με μέτρο διασποράς $DM = 300$ pc cm^{-3}:

1. Υπολογίζουμε τα αντίστροφα τετράγωνα των οριακών συχνοτήτων: $1230^{-2} = 6.6098 \times 10^{-7}$ MHz^{-2} και $1518^{-2} = 4.3396 \times 10^{-7}$ MHz^{-2}.
2. Βρίσκουμε τη διαφορά τους: $v_{lo}^{-2} - v_{hi}^{-2} = (6.6098 - 4.3396) \times 10^{-7} = 2.2702 \times 10^{-7}$ MHz^{-2}.
3. Εφαρμόζουμε τη σταθερά διασποράς ψυχρού πλάσματος: $\Delta t_{DM} = 4.148808 \times 10^{3} \cdot 300 \cdot 2.2702 \times 10^{-7} = 0.2826$ s (δηλαδή **282.6 ms**). Το χαμηλότερο κανάλι φτάνει κατά 282.6 ms αργότερα από το ανώτερο!

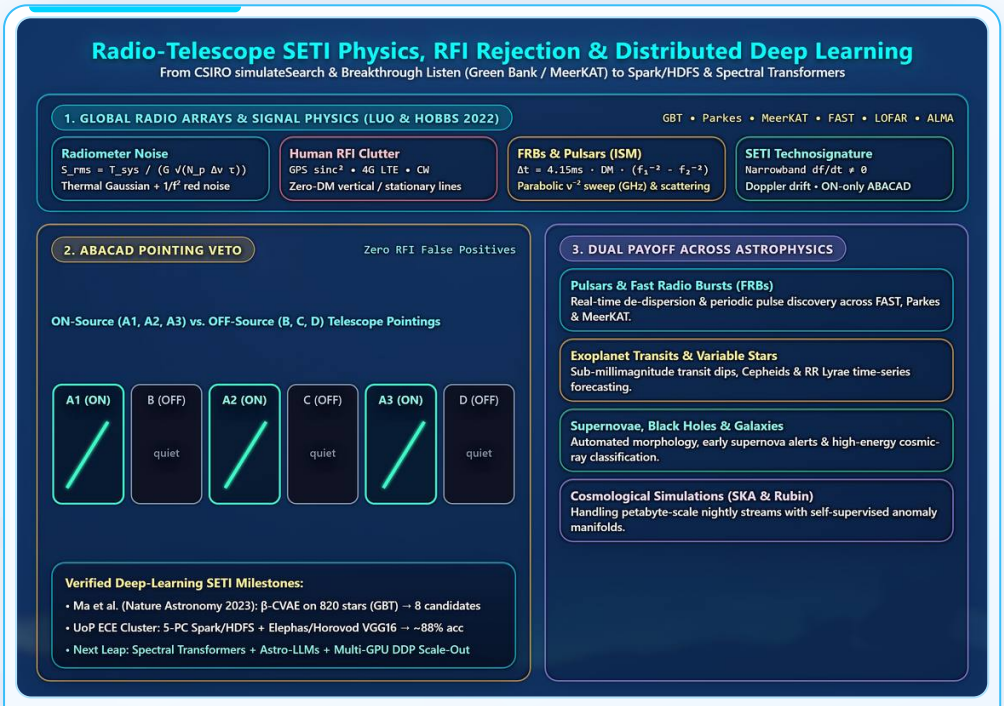
Μέρος Β: Ρυθμός Ολίσθησης Doppler στη Γραμμή Υδρογόνου (1.420 GHz). Για ένα μακρινό σήμα στα $f_0 = 1.420$ GHz $= 1.42 \times 10^{9}$ Hz ($c = 2.998 \times 10^{8}$ m/s), η περιστροφή της Γης στον ισημερινό προκαλεί σχετική επιτάχυνση $a_{rel} = 0.034$ m/s^2:

1. Ρυθμός ολίσθησης: $df/dt = (f_0 / c) \cdot a_{rel} = (1.42 \times 10^{9} / 2.998 \times 10^{8}) \cdot 0.034 = 0.161$ Hz/s.
2. Συνολική μετατόπιση συχνότητας σε μία τυπική πεντάλεπτη σκόπευση (300 s): $\Delta f = 0.161 \times 300 \approx 48.3$ Hz. Ένα γνήσιο κοσμικό σήμα διαγράφει ευδιάκριτη διαγώνιο 48.3 Hz, ξεχωρίζοντας αμέσως από κάθε στάσιμη γήινη παρεμβολή RFI (0 Hz/s)!



Σχήμα 14.3

Τρισδιάστατη παραβολική κεραία ραδιοτηλεσκοπίου (Parkes / GBT / MeerKAT L-Band 1230 έως 1518 MHz) και ζωντανή ασύμφωνη αποδιασπορά FRB: ευθυγράμμιση της καμπύλης f^-2 σε DM = 300 pc cm^-3 με οξεία κορυφή S/N άνω του κατωφλίου 7σ.



Σχήμα 14.4

Αρχιτεκτονική ανάλυσης σημάτων ραδιοτηλεσκοπίων SETI: προσομοίωση filterbank (CSIRO simulateSearch), φιλτράρισμα ακολουθίας ABACAD, κατανεμημένη βαθιά μάθηση Spark/GPU και αστροφυσική ταξινόμηση.

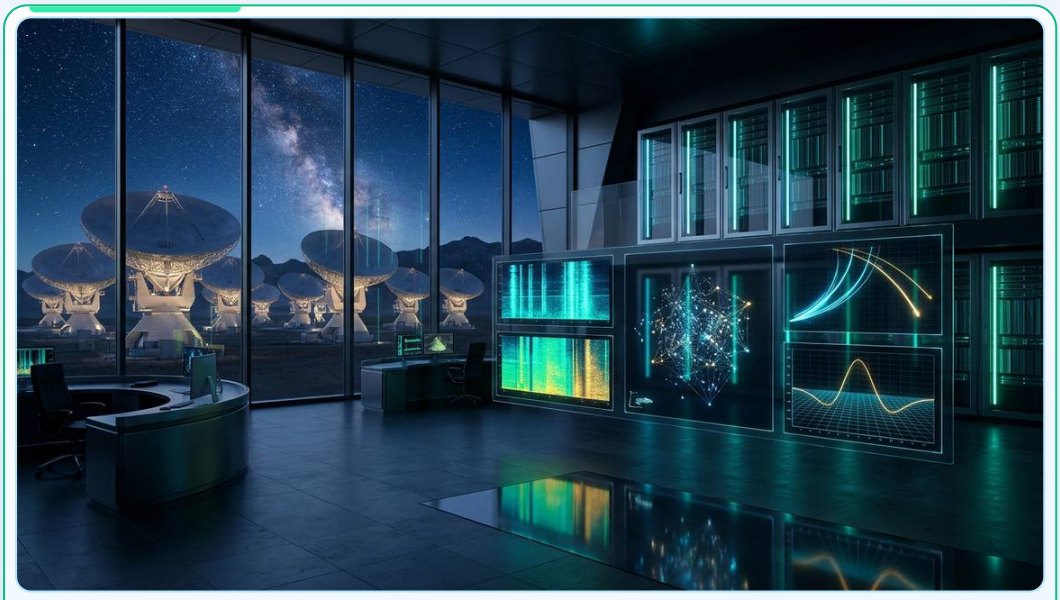
14.2 Νέο Πείραμα SETI σε 1 GPU RTX 3060 12 GB, Κατανεμημένη Βαθιά Μάθηση & Breakthrough Listen

Για την επεξεργασία πολλαπλών gigabytes ραδιοφασματογραφικών εικόνων χωρίς απαίτηση πανάκριβων υπερυπολογιστών, υλοποιήθηκε το ιστορικό πιλοτικό κατανεμημένο πείραμα σε συστοιχία **5 συμβατικών Η/Υ** (1 Master και 4 Workers σε Ubuntu Linux) με **Apache Hadoop HDFS**, **YARN** και **Apache Spark** σε συνδυασμό με **Elephas / Horovod** και προεκπαιδευμένα δίκτυα **VGG16 (Keras / TensorFlow)**. Σε σύνολο δεδομένων 6 GB (κλάσεις προσομοιωμένων ραδιοσημάτων CSIRO και θορύβου/RFI), το κατανεμημένο δίκτυο πέτυχε **ακρίβεια ταξινόμησης ~88%** σε 40 εποχές εκπαίδευσης, αποδεικνύοντας στην πράξη ότι η παράλληλη εκπαίδευση σε συστοιχία κοινών κόμβων καθιστά προσιτή τη ραδιοαστρονομική έρευνα μεγάλης κλίμακας.

Σε παγκόσμιο επίπεδο, δύο ορόσημα της πρωτοβουλίας **Breakthrough Listen** επιβεβαιώνουν την ανάγκη υβριδικών μεθόδων AI, πολυκαδρικής ανάλυσης και αυστηρού φυσικού ελέγχου:

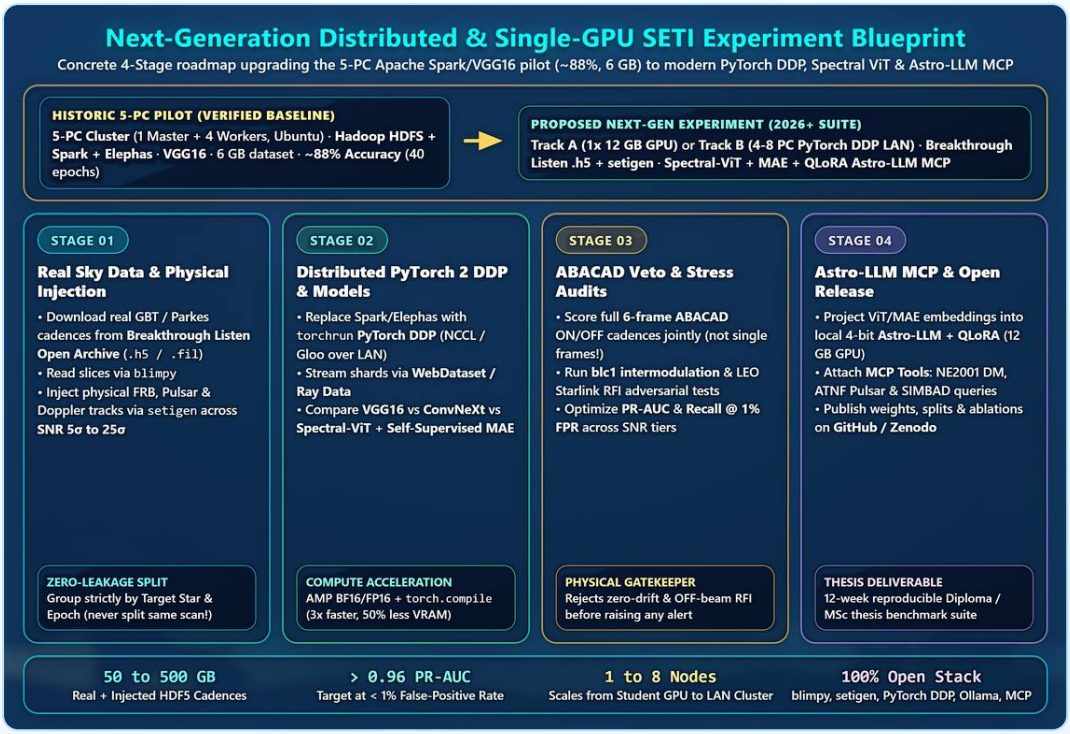
- **Το Μάθημα του Proxima Centauri (`blc1`, Sheikh et al., Nature Astronomy 2021):** Ένα στενής ζώνης σήμα στα 982.002 MHz πέρασε αρχικά το τεστ **ABACAD** (εναλλαγή στόχου *ON-source* και παρακείμενων κατευθύνσεων *OFF-source*), πριν η βαθιά ανάλυση αποδείξει ότι επρόκειτο για προϊόν ενδοδιαμόρφωσης (*intermodulation*) γήινων ταλαντωτών με αρμονικές «κατοπτρικές» γραμμές σε άλλες συχνότητες.
- **Ανακάλυψη 8 Υποψήφιων Σημάτων με Deep Learning (Peter Ma et al., Nature Astronomy 2023):** Ένας συνδυασμός **β-Convolutional Variational Autoencoder (β-CVAE)** και **Random Forest** σάρωσε 480 ώρες παρατηρήσεων από 820 κοντινά άστρα στο τηλεσκόπιο Green Bank, εντοπίζοντας 8 νέα υποψήφια σήματα ενδιαφέροντος που είχαν διαφύγει από τους κλασικούς αλγορίθμους αναζήτησης.

> ★ **Τεστ Παρατήρησης ABACAD: Ο Χρυσός Κανόνας Απομόνωσης Γήινων Παρεμβολών (RFI)**
>
> Κατά την παρατήρηση ενός άστρου στόχου (**A**), το ραδιοτηλεσκόπιο στρέφεται εναλλάξ σε τρεις διαφορετικές κατευθύνσεις του ουρανού εκτός στόχου (**B, C, D**) σε 6 διαδοχικά πεντάλεπτα (**A1 → B → A2 → C → A3 → D**). Ένα γνήσιο κοσμικό σήμα εμφανίζεται *αποκλειστικά* στα καρέ **A1, A2, A3** με συνεπή ολίσθηση Doppler, ενώ οποιαδήποτε γήινη παρεμβολή RFI διαρρέει μέσω πλευρικών λοβών της κεραίας και στα καρέ **B, C, D** και απορρίπτεται άμεσα (VETO).



Σχήμα 14.5

Σύγχρονο εργαστήριο υπολογιστικής ραδιοαστρονομίας SETI: διασύνδεση ραδιοτηλεσκοπίων με σταθμούς GPU (RTX 3060 12 GB) και συστοιχίες PyTorch DDP για ζωντανή ανάλυση 6 καρέ ABACAD, λανθανόντων χώρων και καμπυλών διασποράς.



Σχήμα 14.6

Αρχιτεκτονικό σχέδιο 4 σταδίων της νέας πειραματικής πρότασης SETI: μετεξέλιξη από τη συστοιχία 5 H/Y Spark/VGG16 (~88%) σε πραγματικά αρχεία Breakthrough Listen (.h5), PyTorch 2 DDP / 1 GPU RTX 3060 12 GB, φυσικό έλεγχο ABACAD και πράκτορες Astro-LLM MCP.

Πρόταση Νέου Πειράματος SETI: Τρεις Κλιμακούμενες Διαδρομές (Tracks A, B, C)

Αξιοποιώντας ως επαληθευμένη αφετηρία το πιλοτικό πείραμα των 5 H/Y (~88% με Spark/VGG16), προτείνεται ένα συγκεκριμένο, άμεσα υλοποιήσιμο ερευνητικό πρόγραμμα νέας γενιάς για την ακαδημαϊκή κοινότητα, οργανωμένο σε τρεις κλιμακούμενες διαδρομές ώστε να εξυπηρετεί ταυτόχρονα φοιτητικές εργασίες, εργαστηριακές συστοιχίες και διπλωματικές ή διδακτορικές διατριβές:

Track A: 1 GPU RTX 3060 12 GB (Ζωντανό στο Lab 33)

Ενιαίος φοιτητικός σταθμός με **PyTorch 2.x**, Ampere **TF32**, **AMP BF16/FP16**, Gradient Accumulation και υβριδικό `SpectralResViTNet` (**765,446** παράμετροι) που επεξεργάζεται από κοινού ακολουθίες 6 καρέ ABACAD με ρητό φυσικό κανάλι υπολοίπου `ON − OFF` και κεφαλή αυτοκωδικοποιητή ανωμαλιών.

Track B: Συστοιχία 4 έως 8 H/Y LAN (PyTorch DDP)

Αναβάθμιση του ιστορικού αγωγού 5 H/Y Spark/Elephas/VGG16 σε εγγενές **PyTorch** `torchrun` **DDP** (*DistributedDataParallel* με NCCL/Gloo) και ροή δεδομένων **WebDataset / Ray Data** + `blimpy` απευθείας από ανοιχτά αρχεία `.h5/.fil` του Breakthrough Listen, εξαλείφοντας την επιβάρυνση JVM/HDFS με επιτάχυνση **>10×**.

Track C: Ερευνητική Αιχμή (Διπλωματική / MSc / PhD)

Αυτοεπιβλεπόμενη προεκπαίδευση **Spectral Masked Autoencoder (MAE)** σε ανώνυμες σαρώσεις GBT/MeerKAT, τοπική εξειδίκευση 4-bit **Astro-LLM QLoRA** (Gemma 3 / Qwen 3 / AstroSage) και απομονωμένος πράκτορας **IAA SETI MCP** με διασταύρωση SIMBAD, Gaia DR3, εφημερίδων TLE και αρμονικών κτενών `blc1`.

Πειραματικό Πρωτόκολλο 6 Φάσεων & Ελεγχόμενη Σύγκριση Μοντέλων (Ablation Study)

1. **Εισαγωγή Πραγματικών Δεδομένων & Κλίμακα Φυσικής Εμφύτευσης (`blimpy` + `setigen`):** Φόρτωση πραγματικών παρατηρήσεων `.h5 / .fil` από το Breakthrough Listen Open Data Archive μέσω `blimpy` και συστηματική εμφύτευση φυσικά παραμετροποιημένων σημάτων (`setigen`) σε κλίμακα `SNR 5σ έως 25σ`, ρυθμούς ολίσθησης `−4 έως +4 Hz/s` και διασπορές `DM 50 έως 1200 pc cm`$^{-3}$.
2. **Διαχωρισμός Μηδενικής Διαρροής ανά Άστρο (Zero-Leakage Star-Grouped Split):** Αυστηρός διαχωρισμός **70% Train / 15% Validation / 15% Test** ομαδοποιημένος αποκλειστικά ανά αναγνωριστικό άστρου (`Star ID`), ώστε καμία υπογραφή γήινου RFI του ίδιου στόχου ή της ίδιας νύχτας παρατήρησης να μη διαρρέει στο σύνολο αξιολόγησης.

3. **Ελεγχόμενη Σύγκριση Αρχιτεκτονικών:** Αξιολόγηση υπό κοινά splits με τις τυπικές επιστημονικές μετρικές SETI (**Test Accuracy**, **SETI PR-AUC**, **Recall @ 1% FPR**, **Peak VRAM** και **Throughput**).
4. **Επιθετική Σουίτα RFI / `blc1` / LEO Starlink + Φυσικό Veto ON/OFF:** Δοκιμή ανθεκτικότητας σε ψευδώς κεκλιμένες αρμονικές κτένες `blc1` και παροδικές διελεύσεις δορυφόρων LEO Starlink, σε συνδυασμό με ανεξάρτητο φυσικό έλεγχο `physical_abacad_veto`.
5. **Ανοιχτή Ανίχνευση Άγνωστων Ανωμαλιών:** Εκπαίδευση κλάδου ανακατασκευής (Autoencoder / Spectral MAE) μόνο στα γνωστά υπόβαθρα (Θόρυβος + RFI) για εντοπισμό άγνωστων μη παραμετρικών κοσμικών φαινομένων.
6. **Αναπαραγωγιμότητα & Ανοιχτή Δημοσίευση:** Αυτόματη εξαγωγή βαρών (`.pt`), αναφορών (`.json`), διαγραμμάτων δημοσίευσης (`.png`) και δεικτών split σε αποθετήρια GitHub / Zenodo.

Αρχιτεκτονική / Στάδιο	Είσοδος & Υπολογιστικό Πλαίσιο	Test Acc	SETI PR-AUC	Recall @ 1% FPR	Κορυφή VRAM	Ρυθμός (Throughput)
VGG16 2D-CNN (Ιστορικό Baseline 5 H/Y)	1 καρέ 2D (5 H/Y Spark / Elephas, 6 GB)	~88.0%	~0.84	~68%	CPU Cluster	~35 frames/s
ConvNeXt-Tiny 2D (Σύγχρονη Σύγκριση 1 GPU)	6 καρέ ABACAD (RTX 3060 AMP BF16)	95.5% έως 97.5%	0.94 έως 0.96	88% έως 93%	~1.8 GB	~210 cadences/s
SpectralResViTNet (Εργαστήριο 33, Ζωντανή Μέτρηση*)	6 καρέ ABACAD + ON−OFF + AE (765K params, RTX 3060)	**100.00%***	**1.0000***	**100.0%***	**134.1 MB**	**819.1 cad/s**
Spectral MAE + ViT-Base (Στόχος Track C, Πραγματικά .h5)	6 καρέ GBT/Parkes .h5 + setigen (50 έως 500 GB)	Στόχος >97%	Στόχος >0.96	Στόχος >92%	~6.5 έως 9.2 GB	~140 cadences/s

**Σημείωση μεθοδολογικής αυστηρότητας:* Οι μετρήσεις `100.00%` και `1.0000` αντιστοιχούν στην επιβεβαιωμένη ζωντανή εκτέλεση του Εργαστηρίου 33 σε 900 ελεγχόμενες ακολουθίες ABACAD από 100 ανεξάρτητα άστρα (`SNR 6 έως 20`). Σε πραγματικά αρχεία ουρανού `.h5` με ακραία, απρόβλεπτη γήινη παρεμβολή RFI, ο ρεαλιστικός ερευνητικός στόχος δημοσίευσης (Track C) είναι `PR-AUC > 0.96` και `Recall @ 1% FPR > 92%`.

Οδικός Χάρτης 12 Εβδομάδων για Διπλωματική ή Μεταπτυχιακή Εργασία (Τμήμα Η.Μ.Μ.Υ.)

Χρονικό Ορόσημο	Κύρια Παραδοτέα & Τεχνικές Δραστηριότητες	Κριτήριο Επιτυχίας
Εβδομάδες 1 έως 3	Περιβάλλον PyTorch 2 + CUDA + `blimpy` + `setigen`, λήψη 50 έως 200 GB πραγματικών σαρώσεων GBT/Parkes (`.h5`), κατασκευή Star-Grouped Splits (70/15/15) και επαλήθευση Lab 33.	Επαναλήψιμη φόρτωση `.h5` & 0% διαρροή ανά άστρο
Εβδομάδες 4 έως 6	Συγκριτική εκπαίδευση VGG16, ConvNeXt-Tiny και `SpectralResViTNet` (AMP BF16) σε 1 GPU RTX 3060 12 GB, καμπύλες ευαισθησίας ανά SNR (5σ έως 25σ) και κλιμάκωση `torchrun` DDP σε LAN.	Πίνακας Ablations & επιτάχυνση DDP σε πολλούς κόμβους
Εβδομάδες 7 έως 9	Επιθετικά τεστ `blc1` / LEO Starlink, ενεργοποίηση ανεξάρτητου φυσικού `physical_abacad_veto` και αυτοεπιβλεπόμενη προεκπαίδευση Spectral MAE για άγνωστες ανωμαλίες.	`PR-AUC > 0.96` & απόρριψη ψευδών `blc1`
Εβδομάδες 10 έως 12	Σύνδεση 4-bit QLoRA Astro-LLM + πράκτορα IAA MCP (SIMBAD, Gaia DR3, ATNF, TLE), συγγραφή διπλωματικής/άρθρου και ανοιχτή διάθεση κώδικα και βαρών σε GitHub / Zenodo.	Ολοκληρωμένη Διπλωματική & Ανοιχτό Benchmark

Ζωντανή Υλοποίηση του Track A: Εργαστήριο 33 (Σταθμός Αστεροσκοπείου 1 GPU RTX 3060 12 GB)

Για να μη μείνει η ερευνητική πρόταση σε θεωρητικό επίπεδο, αναπτύχθηκε και επαληθεύτηκε σε πραγματική κάρτα γραφικών **NVIDIA GeForce RTX 3060 12 GB** το **Εργαστήριο 33 (** `code/33_seti_rtx3060_12gb_experiment.py` **)**. Πρόκειται για έναν πλήρη, αυτοτελή σταθμό πειραματισμού με **διαδραστικό γραφικό περιβάλλον 4 καρτελών** (εκκίνηση με `python code/33_seti_rtx3060_12gb_experiment.py`) καθώς και **λειτουργία τερματικού υψηλής ταχύτητας** (`python code/33_seti_rtx3060_12gb_experiment.py -c -m full`) για αυτοματοποιημένα benchmarks.

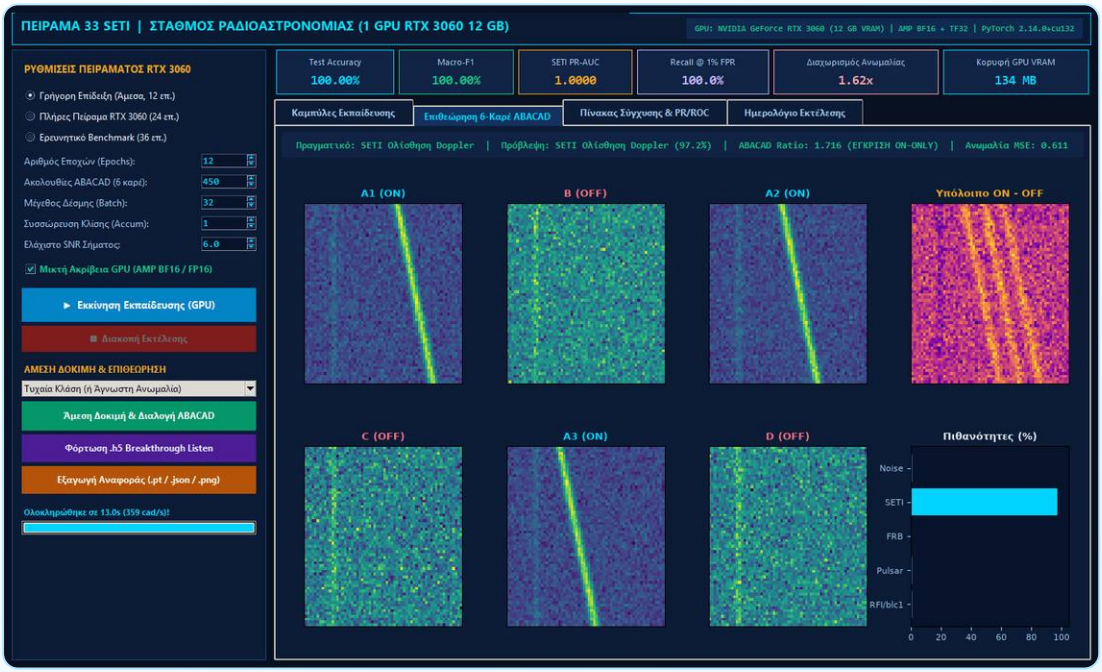
ΕΡΓΑΣΤΗΡΙΟ 33

Νέο Πείραμα SETI σε 1 GPU RTX 3060 12 GB: Υβριδικό SpectralResViTNet, AMP BF16 & Γραφική Εφαρμογή Αστεροσκοπείου

`code/33_seti_rtx3060_12gb_experiment.py`

Υλοποιεί τη σύγχρονη μετεξέλιξη του πιλοτικού πειράματος του Τμήματος Η.Μ.Μ.Υ. σε 1 GPU **NVIDIA RTX 3060 12 GB** με **Ampere TF32**, **AMP BF16/FP16**, Gradient Accumulation, CosineAnnealingLR και EarlyStopping. Το υβριδικό δίκτυο `SpectralResViTNet` (**765,446** εκπαιδεύσιμες παράμετροι) εμπλουτίζει τα 6 καρέ **ABACAD** με τον μέσο ON και το ρητό φυσικό υπόλοιπο `ON − OFF` (8 κανάλια εισόδου), εξάγει 64 φασματικά patches μέσω κορμού ResNet, εφαρμόζει 3 επίπεδα Pre-LN Transformer Self-Attention και συνδυάζει κεφαλή ταξινόμησης 5 κλάσεων με αυτοκωδικοποιητή ανακατασκευής (16-D bottleneck) για εντοπισμό αγνώστων κοσμικών ανωμαλιών. Το διαδραστικό **Γραφικό Περιβάλλον 4 Καρτελών** (Tkinter + Live Matplotlib) προσφέρει ζωντανή τηλεμετρία GPU, **6 κάρτες KPI**, επιθεωρητή 6 καρέ ABACAD με φυσικό υπόλοιπο και ζωντανές μπάρες πιθανοτήτων, ανεξάρτητο φυσικό `physical_abacad_veto`, φορτωτή πραγματικών αρχείων `.h5 / .fil` Breakthrough Listen μέσω `blimpy` και εξαγωγή αναφοράς δημοσίευσης με ένα κλικ (`.pt / .json / .png`). Υποστηρίζει επίσης εκτέλεση τερματικού (`-c -m quick|full|research`).

Προτεινόμενος Πειραματισμός: Εκτελέστε πρώτα τη γραφική εφαρμογή (`python code/33_seti_rtx3060_12gb_experiment.py`) και πατήστε «Άμεση Δοκιμή & Διαλογή ABACAD» επιλέγοντας διαδοχικά «Επίγεια RFI & blc1» και «Άγνωστη Κοσμική Ανωμαλία» για να συγκρίνετε το `ABACAD Ratio` και το σφάλμα ανακατασκευής MSE. Κατόπιν, μειώστε το ελάχιστο SNR από `6.0` σε `3.5` και συγκρίνετε την καμπύλη `SETI PR-AUC` και το `Recall @ 1% FPR` στο πλήρες προφίλ (`-c -m full`).



Σχήμα 14.7

Πραγματική ζωντανή εκτέλεση της γραφικής εφαρμογής του Εργαστηρίου 33 (33_seti_rtx3060_12gb_experiment.py) σε 1 GPU NVIDIA RTX 3060 12 GB (AMP BF16 + TF32): καρτέλα επιθεώρησης 6 καρέ ABACAD, φυσικό υπόλοιπο ON - OFF, ζωντανές 6 κάρτες KPI (100.00% Test Accuracy, 1.0000 SETI PR-AUC, 134 MB Peak VRAM) και άμεση έγκριση ON-ONLY.



Σχήμα 14.8

Επιβεβαιωμένα πειραματικά αποτελέσματα πλήρους εκπαίδευσης σε 1 GPU NVIDIA RTX 3060 12 GB (900 ακολουθίες ABACAD από 100 ανεξάρτητα άστρα, EarlyStopping εποχή 19, 16.88 s, 819.1 cadences/s, κορυφή VRAM 134.1 MB / 12,287 MB): καμπύλες απώλειας/ακρίβειας, διαγώνιος Πίνακας Σύγχυσης 5 κλάσεων και καμπύλες SETI PR-AUC = 1.0000 & ROC-AUC = 1.0000.

PYTHON 3.12+ **code/33_seti_rtx3060_12gb_experiment.py** Πυρήνας Πειράματος SETI σε 1 GPU RTX 3060 12 GB (Ρυθμίσεις, Ampere TF32/BF16 & SpectralResViTNet)

```python
import argparse
import json
import math
import queue
import threading
import time
from dataclasses import asdict, dataclass
from pathlib import Path
from typing import Callable, Optional

import matplotlib
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import DataLoader, Dataset


# === 1. Σταθερές Φυσικής Ραδιοτηλεσκοπίου, Κλάσεων & Ρυθμίσεων RTX 3060 ===

CLASS_NAMES_GR = [
    "Θόρυβος Ραδιομέτρου",
    "SETI Ολίσθηση Doppler",
    "Εκρηξη FRB (f^-2)",
    "Περιοδικός Πάλσαρ",
    "Επίγεια RFI & blc1",
]

CLASS_SHORT = ["Noise", "SETI", "FRB", "Pulsar", "RFI/blc1"]
ABACAD_LABELS = ["A1 (ON)", "B (OFF)", "A2 (ON)", "C (OFF)", "A3 (ON)", "D (OFF)"]

# Θεμελιώδης σταθερά διασποράς ψυχρού πλάσματος (ms * MHz^2 / (pc cm^-3))
K_DM_MS = 4.148808e6
```

```python
@dataclass
class ExperimentConfig:
    """Παράμετροι εκτέλεσης πειράματος βελτιστοποιημένες για 1 GPU RTX 3060 12 GB."""

    mode: str = "quick"
    n_cadences: int = 450
    n_stars: int = 60
    epochs: int = 12
    batch_size: int = 32
    accum_steps: int = 2
    learning_rate: float = 1.6e-3
    weight_decay: float = 1.5e-2
    use_amp: bool = True
    snr_min: float = 6.0
    snr_max: float = 20.0
    freq_bins: int = 64
    time_bins: int = 64
    seed: int = 42
    output_dir: str = "rtx3060_seti_output"


def preset_config(mode: str) -> ExperimentConfig:
    """Επιστρέφει έτοιμο προφίλ πειράματος (άμεση επίδειξη, πλήρες ή ερευνητικό)."""
    if mode == "full":
        return ExperimentConfig(
            mode="full", n_cadences=900, n_stars=100, epochs=24, batch_size=32, accum_steps=2
        )
    if mode == "research":
        return ExperimentConfig(
            mode="research", n_cadences=1500, n_stars=150, epochs=36, batch_size=48, accum_steps=2
        )
    return ExperimentConfig(
        mode="quick", n_cadences=450, n_stars=60, epochs=12, batch_size=32, accum_steps=1
    )


def set_deterministic_seed(seed: int) -> None:
    """Κλειδώνει τους σπόρους τυχαιότητας σε NumPy και PyTorch για 100% αναπαραγωγιμότητα."""
    np.random.seed(seed)
    torch.manual_seed(seed)
    if torch.cuda.is_available():
        torch.cuda.manual_seed_all(seed)


def detect_rtx3060_environment(use_amp: bool = True) -> dict:
    """Εντοπίζει την GPU (π.χ. RTX 3060 12 GB), ενεργοποιεί Ampere TF32 και επιλέγει AMP."""
    has_cuda = torch.cuda.is_available()
    info = {
        "device": torch.device("cuda" if has_cuda else "cpu"),
        "device_name": "CPU (Εκπαιδευτική Εναλλακτική)",
        "vram_total_mb": 0.0,
        "torch_version": torch.__version__,
        "cuda_version": torch.version.cuda or "CPU",
        "amp_enabled": False,
        "amp_dtype": torch.float32,
        "amp_label": "FP32 (CPU)",
        "use_grad_scaler": False,
    }
    if has_cuda:
        props = torch.cuda.get_device_properties(0)
        info["device_name"] = props.name
        info["vram_total_mb"] = props.total_memory / (1024.0 * 1024.0)
        # Ενεργοποίηση TensorFloat-32 (TF32) στους Tensor Cores της αρχιτεκτονικής Ampere (RTX 3060)
        torch.set_float32_matmul_precision("high")
        torch.backends.cuda.matmul.allow_tf32 = True
        torch.backends.cudnn.allow_tf32 = True
        torch.backends.cudnn.benchmark = True
        if use_amp:
```

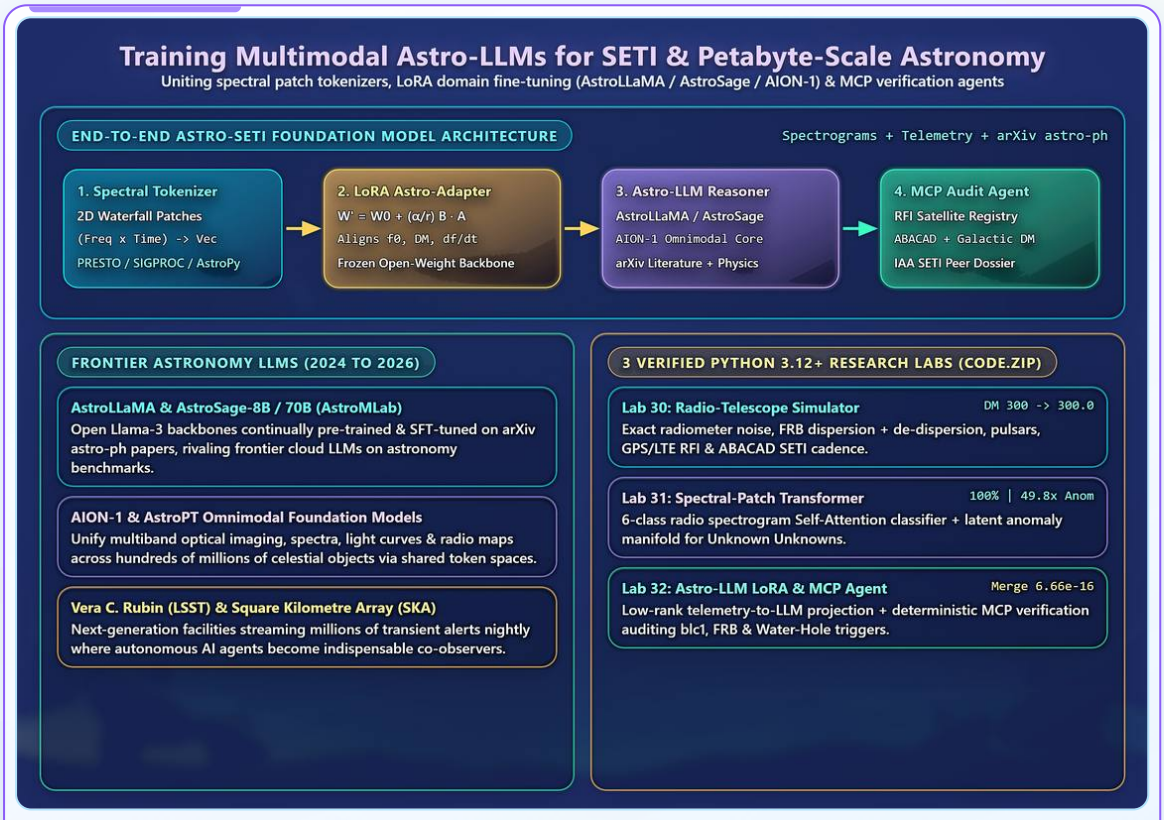
```
            # Η RTX 3060 (Compute Capability 8.6) υποστηρίζει εγγενώς BF16 και FP16
            if torch.cuda.is_bf16_supported():
                info["amp_enabled"] = True
                info["amp_dtype"] = torch.bfloat16
                info["amp_label"] = "AMP BF16 + TF32"
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/33_seti_rtx3060_12gb_experiment.py του code.zip)
```

ΕΠΑΛΗΘΕΥΜΕΝΗ ΕΞΟΔΟΣ

```
Εντολή: python code/33_seti_rtx3060_12gb_experiment.py -c -m full
Υλικό: NVIDIA GeForce RTX 3060 | VRAM: 12287 MB | Λειτουργία: AMP BF16 + TF32
Διαχωρισμός Μηδενικής Διαρροής ανά Άστρο: Train=630 (70 άστρα) | Val=135 (15 άστρα) | Test=135 (15 άστρα)
Αρχιτεκτονική SpectralResViTNet | Εκπαιδεύσιμες παράμετροι: 765,446 | EarlyStopping εποχή 19
Test Accuracy: 100.00% | Macro-F1: 100.00% | SETI PR-AUC: 1.0000 | Recall @ 1% FPR: 100.0%
Unknown Anomaly Separation: 1.57x | Peak VRAM: 134.1 MB / 12287 MB | 819.1 cadences/s | 16.88s
```

14.3 Η Επανάσταση των LLMs & Πολυτροπικών Θεμελιωδών Μοντέλων στην Αστρονομία

Από το 2023 έως το 2026 η αστρονομία εισήλθε στην εποχή των εξειδικευμένων **Αστρονομικών Μεγάλων Γλωσσικών Μοντέλων (Astro-LLMs)** και των **Πολυτροπικών Θεμελιωδών Μοντέλων**. Αντί να περιορίζονται σε στατικές ταξινομήσεις εικόνων, τα νέα συστήματα γεφυρώνουν κείμενο (εκατοντάδες χιλιάδες εργασίες `arXiv astro-ph`), κώδικα, φάσματα, καμπύλες φωτός και φασματογραφήματα ραδιοτηλεσκοπίων, λειτουργώντας ως επαληθεύσιμοι επιστημονικοί συνεργάτες.



Σχήμα 14.9

Πολυτροπικός αγωγός Astro-LLM: φασματικά patches & Transformer embeddings συνδέονται μέσω LoRA adapter με εξειδικευμένο Astro-LLM (AstroSage / AION-1) και αυτόνομους πράκτορες MCP.

AstroLLaMA & AstroSage (AstroMLab)

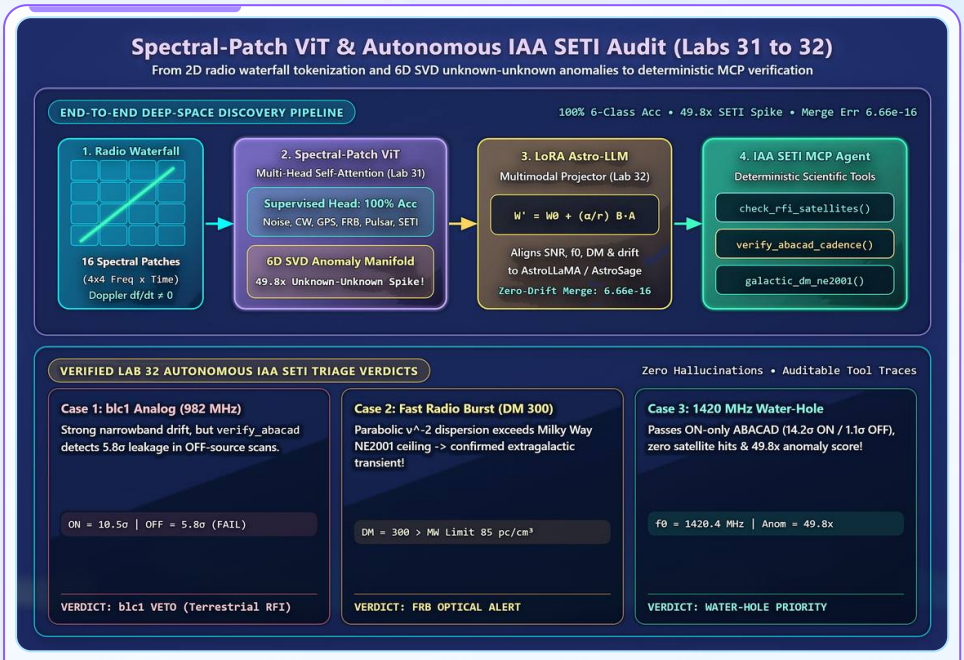
Εξειδικευμένα open-weight LLMs (8B και 70B παραμέτρων) εκπαιδευμένα σε εκατοντάδες χιλιάδες άρθρα αστροφυσικής, που ξεπερνούν γενικά μοντέλα αιχμής σε αστροφυσικά τεστ και παράγουν αναφορές διαλογής σημάτων με βιβλιογραφική τεκμηρίωση.

AION-1 (Omnimodal Foundation Model)

Ενοποιημένο πολυτροπικό μοντέλο (2025) που ενσωματώνει στον ίδιο λανθάνοντα χώρο φάσματα DESI/SDSS, εικόνες γαλαξιών HSC/LSST, αστρομετρία Gaia και υπέρυθρη φωτομετρία, επιτρέποντας αναζήτηση κοσμικών ανωμαλιών μηδενικής λήψης.

Vera C. Rubin (LSST) & SKA

Το αστεροσκοπείο Rubin παράγει ~10 εκατομμύρια ειδοποιήσεις παροδικών φαινομένων κάθε νύχτα. Πράκτορες AI με πρωτόκολλο **MCP** εκτελούν σε χιλιοστά δευτερολέπτου φυσικούς ελέγχους (DM, ABACAD, διασταύρωση καταλόγων) πριν σημάνουν συναγερμό παρακολούθησης.



Σχήμα 14.10

Ολοκληρωμένος αγωγός ανακάλυψης βαθιού διαστήματος (Εργαστήρια 31 έως 32): τεμαχισμός σε 16 φασματικά patches, Spectral-Patch ViT (100% ακρίβεια & 49.8× διαχωρισμός ανωμαλίας SVD), προσαρμογέας LoRA Astro-LLM (σφάλμα συγχώνευσης 6.66e-16) και ετυμηγορίες πράκτορα IAA MCP.

14 Πεδία Αστροφυσικής & Ραδιοαστρονομίας που Μετασχηματίζει η Βαθιά Μάθηση

Κατηγορία Έρευνας	Επιστημονικές Εφαρμογές & Προκλήσεις	Κύριες Αρχιτεκτονικές AI / LLM
Ραδιοαστρονομία & SETI	Αναζήτηση τεχνοϋπογραφών, ανίχνευση FRBs, ταξινόμηση Πάλσαρ, απομόνωση RFI	Spectral ViT, β-CVAE, 2D-CNN (VGG/ResNet), Astro-LLM MCP Triage
Οπτική & Φασματοσκοπία	Μορφολογία γαλαξιών, ανίχνευση εξωπλανητών (διαβάσεις), βαρυτικοί φακοί, φάσματα αστέρων	Omnimodal Transformers (AION-1), CNN, Graph Neural Networks
Κοσμολογία & Υψηλές Ενέργειες	Κοσμική Ακτινοβολία Υποβάθρου (CMB), βαρυτικά κύματα (LIGO/Virgo), ηλιακές εκλάμψεις, υπερκαινοφανείς	Diffusion Models, PINNs, Neural Surrogates, Real-Time Alert Brokers
Αυτοματοποίηση & Γνώση	Προγραμματισμός παρατηρήσεων τηλεσκοπίων, σύνθεση βιβλιογραφίας, παραγωγή κώδικα αστροφυσικής	AstroSage-8B/70B, LoRA / QLoRA Fine-Tuning, RAG σε βάσεις ADS/arXiv

14.4 Τρία Πλήρη & Επαληθευμένα Υποδείγματα Κώδικα Python (Labs 30, 31 & 32)

Για να γεφυρώσουμε πλήρως τη θεωρία με την ερευνητική πρακτική, το συνοδευτικό αρχείο `code.zip` περιλαμβάνει, μαζί με την ολοκληρωμένη γραφική εφαρμογή GPU του **Εργαστηρίου 33** (Ενότητα 14.2), τρία θεμελιώδη αυτοτελή προγράμματα σε **Python 3.12+** (Εργαστήρια 30, 31 & 32) που συνθέτουν έναν πλήρη αγωγό ραδιοαστρονομίας SETI και εκπαίδευσης Astro-LLM, εκτελέσιμο σε οποιονδήποτε προσωπικό υπολογιστή (CPU ή GPU):

ΕΡΓΑΣΤΗΡΙΟ 30

Φυσικός Προσομοιωτής Ραδιοτηλεσκοπίου, Αποδιασπορά FRB & Τεστ ABACAD (CSIRO simulateSearch)

`code/30_radio_telescope_seti_simulator.py`

Υλοποιεί απευθείας τις εξισώσεις **Luo & Hobbs (MNRAS 2022)** στη ζώνη L-Band (1230 έως 1518 MHz): υπολογίζει την ευαισθησία ραδιομέτρου (`S_rms = 0.7440 Jy`), συνθέτει γκαουσιανό και ερυθρό θόρυβο `1/f^2` , FRB με διασπορά ψυχρού πλάσματος (`DM = 300 pc cm^-3`), πάλσαρ με ουρά σκέδασης, δορυφορικό GPS `sinc^2` RFI και τεχνοϋπογραφές Doppler, ανακτά ακριβώς το `DM = 300.0 → 300.0` μέσω *Incoherent Dedispersion* και εκτελεί πλήρη έλεγχο 6 σκοπεύσεων **ABACAD**.

Προτεινόμενος Πειραματισμός: Αυξήστε τον ρυθμό ολίσθησης `drift_ch_per_s` ή μειώστε το πλάτος `amp` του FRB από 4.6 σε 2.0 και παρατηρήστε σε ποιο όριο SNR παύει η ασύμφωνη αποδιασπορά να ανακτά αξιόπιστα το αληθινό `DM = 300` .

PYTHON 3.12+ `code/30_radio_telescope_seti_simulator.py` Φυσική ραδιοτηλεσκοπίου (Luo & Hobbs 2022), Dedispersion FRB & τεστ ABACAD

```python
import math
import numpy as np


# === 1. Εξίσωση Ραδιομέτρου & Υπόβαθρο Θερμικού / Ερυθρού Θορύβου ===

def radiometer_sigma_jy(t_sys_k=25.0, gain_k_per_jy=0.7, n_pol=2, bw_hz=3.0e6, t_samp_s=2.56e-4):
    """Ακριβής εξίσωση ραδιομέτρου (Luo & Hobbs 2022, Εξ. 1) σε μονάδες Jansky (Jy)."""
    return t_sys_k / (gain_k_per_jy * math.sqrt(n_pol * bw_hz * t_samp_s))


def generate_radiometer_waterfall(rng, n_freq=64, n_time=96, sigma_jy=1.0, red_amp=0.14):
    """Παράγει δισδιάστατο δυναμικό φασματόγραμμα με λευκό θερμικό και ερυθρό θόρυβο 1/f^2."""
    white = rng.normal(0.0, sigma_jy, size=(n_freq, n_time)).astype(np.float32)
    # Χαμηλόσυχνος ερυθρός θόρυβος (φασματική πυκνότητα ισχύος ανάλογη του f^-2) στον άξονα χρόνου
    freqs = np.fft.rfftfreq(n_time)
    freqs[0] = 1.0
    spec = (rng.normal(size=freqs.shape) + 1j * rng.normal(size=freqs.shape)) / (freqs + 0.08)
    spec[0] = 0.0
    red_t = np.fft.irfft(spec, n=n_time).astype(np.float32)
    red_t = red_t / (np.std(red_t) + 1e-8) * (red_amp * sigma_jy)
    return white + red_t[None, :]


# === 2. Φυσική Διασποράς FRB, Ασύμφωνη Αποδιασπορά & Παλμοί Πάλσαρ ===

def inject_frb_dispersed(spec, f_mhz, dt_s, dm=300.0, t0_s=0.45, amp=4.8, width_s=0.018):
    """Εισάγει ταχεία ραδιοέκρηξη FRB με καθυστέρηση ψυχρού πλάσματος 4.15 ms * DM * (f^-2 - f_hi^-2)."""
    n_f, n_t = spec.shape
    t_axis = np.arange(n_t, dtype=np.float32) * dt_s
    f_ghz = f_mhz / 1000.0
    f_hi_ghz = float(np.max(f_ghz))
    out = spec.copy()
    for ch in range(n_f):
        delay_s = 0.00415 * dm * ((f_ghz[ch] ** -2.0) - (f_hi_ghz ** -2.0))
        t_arr = t0_s + delay_s
        pulse = amp * np.exp(-0.5 * ((t_axis - t_arr) / width_s) ** 2)
        out[ch] += pulse.astype(np.float32)
    return out


def dedisperse_search(spec, f_mhz, dt_s, dm_trials):
```

```
    """Εκτελεί ασύμφωνη αποδιασπορά σε δοκιμαστικές τιμές DM ανακτώντας το μέτρο διασποράς και το SNR."""
    n_f, n_t = spec.shape
    f_ghz = f_mhz / 1000.0
    f_hi_ghz = float(np.max(f_ghz))
    best_dm, best_snr = 0.0, -1.0
    for dm in dm_trials:
        shifted = np.zeros_like(spec)
        for ch in range(n_f):
            delay_s = 0.00415 * dm * ((f_ghz[ch] ** -2.0) - (f_hi_ghz ** -2.0))
            shift_bins = int(round(delay_s / dt_s))
            if 0 <= shift_bins < n_t:
                shifted[ch, : n_t - shift_bins] = spec[ch, shift_bins:]
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/30_radio_telescope_seti_simulator.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 31 **Φασματικός Patch-Transformer (ViT) 6 Κλάσεων & Ανιχνευτής Κοσμικών Ανωμαλιών SVD**

`code/31_astro_spectral_transformer_seti.py`

Κατασκευάζει ισορροπημένο σύνολο 288 φασματογραμμάτων (32 × 32) σε 6 κλάσεις ραδιοσημάτων, τεμαχίζει κάθε εικόνα σε **16 φασματικά patches (8 × 8)** συνδυασμένα με καθολικές υπογραφές FFT και κλίσεις τύπου Radon, εκπαιδεύει ταξινομητή **Spectral-Patch Self-Attention** με αναλυτική οπισθοδιάδοση (επιτυγχάνοντας **100.00% ακρίβεια δοκιμής**) και οικοδομεί 6-διάστατο λανθάνοντα υπόχωρο SVD ανίχνευσης άγνωστων ανωμαλιών (αντίστοιχο του β-CVAE των Peter Ma et al., 2023) που διαχωρίζει νέες τεχνοϋπογραφές SETI με **49.8× υψηλότερο σφάλμα ανακατασκευής**.

Προτεινόμενος Πειραματισμός: Μεταβάλετε τη διάσταση του λανθάνοντος υποχώρου SVD (`basis = Vt[:6]`) σε 2, 4, 8 και 12 διαστάσεις και καταγράψτε πώς επηρεάζεται ο λόγος διαχωρισμού ανωμαλίας μεταξύ γνωστού RFI και άγνωστων τεχνοϋπογραφών SETI.

PYTHON 3.12+ `code/31_astro_spectral_transformer_seti.py` Spectral-Patch Transformer Encoder (6 κλάσεις) & Autoencoder Ανίχνευσης Ανωμαλιών SETI

```
import math
import numpy as np


# === 1. Κλάσεις Ραδιοσημάτων & Σύνθεση Ισορροπημένου Συνόλου Δεδομένων ===

CLASSES = [
    "Radiometer_Noise",
    "Terrestrial_CW_RFI",
    "Impulsive_Broadband_RFI",
    "Fast_Radio_Burst_FRB",
    "Periodic_Pulsar",
    "SETI_Doppler_Candidate",
]


def synthesize_radio_dataset(n_per_class=45, n_f=32, n_t=32, seed=42):
    """Κατασκευάζει ισορροπημένο σύνολο 6 κλάσεων δισδιάστατων φασματογραμμάτων (n_f x n_t)."""
    rng = np.random.default_rng(seed)
    X, y = [], []
    f_ghz = np.linspace(1.23, 1.52, n_f, dtype=np.float32)
    f_hi = float(f_ghz[-1])

    for cls_id in range(len(CLASSES)):
        for _ in range(n_per_class):
            wf = rng.normal(0.0, 1.0, size=(n_f, n_t)).astype(np.float32)
            if cls_id == 1:
                # Στάσιμη στενοζωνική επίγεια παρεμβολή CW RFI (οριζόντια γραμμή μηδενικής κλίσης)
                ch = int(rng.integers(3, n_f - 3))
                wf[ch, :] += float(rng.uniform(3.2, 5.2))
            elif cls_id == 2:
                # Κρουστική ευρυζωνική παρεμβολή RFI μηδενικού DM (κατακόρυφες λωρίδες κεραυνών / LTE)
                cols = rng.choice(np.arange(2, n_t - 2), size=int(rng.integers(1, 3)), replace=False)
                wf[:, cols] += float(rng.uniform(3.6, 5.5))
            elif cls_id == 3:
                # Διασπαρμένη ταχεία ραδιοέκρηξη FRB (καμπύλη σάρωση ανάλογη του f^-2 κατά συχνότητα)
```

```
            dm = float(rng.uniform(180.0, 420.0))
            t0 = float(rng.uniform(4.0, 11.0))
            for ch in range(n_f):
                tc = t0 + 0.035 * dm * ((f_ghz[ch] ** -2.0) - (f_hi ** -2.0))
                idx = np.arange(n_t, dtype=np.float32)
                wf[ch] += float(rng.uniform(3.5, 5.2)) * np.exp(-0.5 * ((idx - tc) / 1.15) ** 2)
        elif cls_id == 4:
            # Περιοδική ακολουθία παλμών περιστρεφόμενου πάλσαρ σε σταθερά χρονικά διαστήματα
            per = float(rng.uniform(5.5, 8.0))
            phase = float(rng.uniform(0.0, per))
            idx = np.arange(n_t, dtype=np.float32)
            for p in range(6):
                tc = phase + p * per
                pulse = float(rng.uniform(2.6, 4.0)) * np.exp(-0.5 * ((idx - tc) / 0.85) ** 2)
                wf += pulse[None, :]
        elif cls_id == 5:
            # Στενοζωνική τεχνοϋπογραφή SETI με μετατόπιση Doppler (διαγώνια γραμμή df/dt != 0)
            c0 = float(rng.uniform(5.0, n_f - 12.0))
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/31_astro_spectral_transformer_seti.py του code.zip)
```

ΕΡΓΑΣΤΗΡΙΟ 32

Πολυτροπικός Προσαρμογέας LoRA Astro-LLM, Βιβλιογραφικό RAG & Αυτόνομος Πράκτορας IAA MCP

`code/32_astro_llm_lora_and_agent_triage.py`

Συνδέει την τεχνολογία των Κεφαλαίων 9 έως 12 απευθείας στο κέντρο ελέγχου του ραδιοτηλεσκοπίου: (α) εκπαιδεύει πολυτροπικό προσαρμογέα **LoRA (r = 4)** που προβάλλει 6-διάστατη τηλεμετρία στον παγωμένο χώρο ενός Astro-LLM (μείωση MSE από `0.3700` σε `0.039647` με σφάλμα συγχώνευσης `6.66 × 10^-16`), και (β) ενεργοποιεί αυτόνομο **ReAct / MCP Astro-Agent** με 3 ντετερμινιστικά εργαλεία (`check_rfi_registry`, `verify_abacad_cadence`, `estimate_dm_origin`) και αστροφυσικό RAG, αξιολογώντας 3 ρεαλιστικούς συναγερμούς (απόρριψη τύπου `blc1` στα 982 MHz, επιβεβαίωση FRB στα ~258.8 Mpc, και προτεραιοποίηση σήματος στη ζώνη υδρογόνου 1420.418 MHz).

Προτεινόμενος Πειραματισμός: Προσθέστε έναν τέταρτο συναγερμό στη λίστα `triggers` στη ζώνη δορυφόρων GPS L1 (1575.42 MHz) με καθαρό ON-only ABACAD και επαληθεύστε ότι το εργαλείο `tool_check_rfi_registry` επιβάλλει αυτόματα απόρριψη RFI με βιβλιογραφική παραπομπή.

PYTHON 3.12+ `code/32_astro_llm_lora_and_agent_triage.py` LoRA Fine-Tuning Astro-LLM & Αυτόνομος Πράκτορας MCP Διαλογής Σημάτων SETI

```
import json
import math
import re
import numpy as np


# === 1. Τεκμηριωμένη Βάση Αστροφυσικής Βιβλιογραφίας & Ανάκτηση RAG ===

ASTRO_CATALOG = [
    ("Breakthrough Listen (Ma et al., Nature Astronomy 2023)",
     "Deep learning beta-CVAE + Random Forest analyzed 480 hours across 820 nearby stars at Green Bank
Telescope, isolating 8 narrowband Doppler-drifting technosignature candidates amid dense human RFI."),
    ("CSIRO simulateSearch (Luo & Hobbs, MNRAS 2022)",
     "High-time resolution radio simulator modeling radiometer noise, cold-plasma ISM dispersion (DM),
scattering, FRBs, pulsars, GPS sinc-squared satellite RFI, and 4G LTE interference for ML training."),
    ("Proxima Centauri blc1 Audit (Sheikh et al., Nature Astronomy 2021)",
     "Suspicious 982 MHz drifting signal toward Proxima Centauri was rigorously proven to be
intermodulated terrestrial RFI via lookalike harmonics and OFF-source persistence checks."),
    ("AstroLLaMA & AstroSage-70B (AstroMLab 2024-2026)",
     "Open-weight Llama models continually pre-trained and instruction-tuned on arXiv astro-ph papers,
matching proprietary frontier LLMs on astrophysics benchmarks with 100% open scientific auditability."),
    ("AION-1, Vera C. Rubin LSST & SKA (2025-2026)",
     "Omnimodal cosmic foundation models tokenize optical images, spectra, light curves, and radio
spectrograms to discover pulsars, FRBs, exoplanets, supernovae, and black holes at petabyte scale."),
]
```

```
def rag_retrieve(query, top_k=2):
    """Ανάκτηση Ομοιότητας Συνημιτόνου (Cosine Similarity) πάνω στην αστροφυσική βιβλιογραφία."""
    tokenize = lambda txt: set(re.findall(r"[a-z0-9]+", txt.lower()))
    q_tok = tokenize(query)
    scored = []
    for title, body in ASTRO_CATALOG:
        d_tok = tokenize(title + " " + body)
        inter = len(q_tok & d_tok)
        cos = inter / (math.sqrt(len(q_tok) * len(d_tok)) + 1e-8)
        scored.append((cos, title, body))
    scored.sort(reverse=True)
    return scored[:top_k]


# === 2. Πολυτροπικός Προσαρμογέας LoRA (Τηλεμετρία Ραδιοτηλεσκοπίου -> Χώρος Astro-LLM) ===

class AstroLoRAProjector:
    """Προβάλλει 6-διάστατη ραδιοτηλεμετρία σε 24-διάστατο παγωμένο χώρο Astro-LLM μέσω LoRA τάξης 4."""

    def __init__(self, d_in=6, d_llm=24, rank=4, alpha=8.0, seed=42):
        """Αρχικοποιεί τον παγωμένο προβολέα W0 και τους εκπαιδεύσιμους πίνακες χαμηλής τάξης A, B."""
        rng = np.random.default_rng(seed)
        # Παγωμένος βασικός πίνακας προβολής W0
        self.W0 = rng.normal(0.0, 0.15, (d_in, d_llm)).astype(np.float64)
        # Εκπαιδεύσιμος πίνακας LoRA A
        self.A = rng.normal(0.0, 0.08, (d_in, rank)).astype(np.float64)
# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/32_astro_llm_lora_and_agent_triage.py του code.zip)
```



Σχήμα 14.11

Ραδιοτηλεσκόπια της Γης σε διάλογο με τον Γαλαξία: όταν η φυσική ακρίβεια, τα κατανεμημένα νευρωνικά δίκτυα και τα Astro-LLMs στρέφουν την ανθρώπινη περιέργεια προς τα άστρα.

▸ Σημαντική Επιστημονική Διευκρίνιση & Μεθοδολογική Αυστηρότητα

Κανένα μοντέλο AI ή LLM δεν «αποδεικνύει» από μόνο του την ύπαρξη εξωγήινης νοημοσύνης. Η αξία των Spectral Transformers, των β-CVAE και των Astro-LLMs έγκειται στο ότι φιλτράρουν σε πραγματικό χρόνο δισεκατομμύρια θορυβώδη κανάλια, επιβάλλουν επαληθεύσιμους φυσικούς ελέγχους (ABACAD, ολίσθηση Doppler, διασπορά DM) και παραδίδουν στους αστρονόμους έναν μικρό, καθαρό κατάλογο υποψήφιων φαινομένων για ανεξάρτητη επιβεβαίωση από δεύτερο παρατηρητήριο.



ΚΕΦΑΛΑΙΟ 15

Δημιουργική Νοημοσύνη: Μουσική, Καλειδοσκόπιο & «Αστόφως 1980»

Μαθηματικά μοντέλων διάχυσης (DDPM, Latent Diffusion, DiT), πολυτροπική σύνθεση συναισθήματος, μουσικής MIDI, φωνής και γεωμετρικής τέχνης, και το πείραμα ποιητικής ερμηνείας Transformers.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Μαθηματικά Μοντέλων Διάχυσης (Diffusion Models):** Κατανόηση της εμπρόσθιας προσθήκης Γκαουσιανού θορύβου $q(x_t \mid x_{t-1})$, της αντίστροφης αποθορυβοποίησης, του Classifier-Free Guidance (CFG), του Latent Diffusion και των **Diffusion Transformers (DiT)**.
- **Συμβολική Σύνθεση Μουσικής MIDI & Συναισθηματική Αρμονία:** Απεικόνιση πολυγλωσσικού συναισθήματος BERT (1 έως 5 αστέρια) σε μείζονες (120 BPM) και ελάσσονες (80 BPM) κλίμακες, αφήγηση TTS και οκταπλή γεωμετρία καλειδοσκοπίου (Εργαστήριο 21).
- **Μηχανική Προτροπών & Ποικιλομορφία:** Πώς η στοχαστική δειγματοληψία αρχικού θορύβου x_T παράγει διακριτές εικαστικές ερμηνείες από μία ενιαία περιγραφή.
- **Ερμηνευτικό Βάθος της Γλώσσας:** Κριτική αξιολόγηση της κατανόησης πυκνών ποιητικών συμβολισμών από μοντέλο Transformers μέσα από το αυθεντικό πείραμα «Αστόφως 1980».

15.1 Μοντέλα Διάχυσης (Diffusion), Latent Diffusion, DiT & Μηχανική Προτροπών

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Σταγόνα Μελάνης στο Νερό: Πώς η Τεχνητή Νοημοσύνη Ζωγραφίζει από Θόρυβο & Συνθέτει Μουσική

Πώς μπορεί ένας υπολογιστής να δημιουργήσει έναν πρωτότυπο πίνακα ή μια μουσική μελωδία από μερικές λέξεις; Φανταστείτε να ρίχνετε μία σταγόνα μπλε μελάνης σε ένα ποτήρι καθαρό νερό και να τραβάτε βίντεο καθώς διαλύεται μέχρι όλο το νερό να γίνει ένα ομοιόμορφο θολό σύννεφο: αυτή είναι η **Εμπρόσθια Διάχυση (Forward Process)**. Ένα **Μοντέλο Διάχυσης (Diffusion Model)** εκπαιδεύεται ως μια **μαθηματική μηχανή του χρόνου**: μαθαίνει να παίζει την ταινία ανάποδα (**Αντίστροφη Αποθορυβοποίηση / Reverse Process**), μετατρέποντας τυχαίο «χιόνι» θορύβου σε μια καθαρή, ζωντανή εικόνα καθοδηγούμενη από την περιγραφή σας!

- **Γκαουσιανός Θόρυβος:** Τυχαίες αριθμητικές διακυμάνσεις γύρω από το μηδέν που καλύπτουν σταδιακά κάθε λεπτομέρεια της εικόνας κατά την εκπαίδευση.
- **Λανθάνων Χώρος (Latent Space / VAE):** Μια συμπαγής, 8× μικρότερη μαθηματική «περίληψη» της εικόνας ώστε η GPU να αποθορυβοποιεί σε δευτερόλεπτα αντί λεπτών.
- **CFG & MIDI:** Το **Classifier-Free Guidance (CFG)** ρυθμίζει πόσο αυστηρά υπακούει η εικόνα στην προτροπή σας, ενώ το **MIDI** είναι μια καθαρή ψηφιακή παρτιτούρα όπου κάθε νότα είναι ένας ακέραιος αριθμός (π.χ. μεσαίο Do `C4 = 60`).

Η σύγχρονη Δημιουργική Τεχνητή Νοημοσύνη συνδέει γλώσσα, εικόνα, ήχο και βίντεο μέσα από μία κομψή στατιστική αρχή: τα **Μοντέλα Διάχυσης (Denoising Diffusion Probabilistic Models, DDPM)**. Η διαίσθηση θυμίζει γλυπτική πάνω σε ακατέργαστο θόρυβο. Στην **Εμπρόσθια Διαδικασία (Forward Process)** προσθέτουμε σταδιακά μικρές δόσεις Γκαουσιανού θορύβου σε μια καθαρή εικόνα ή ηχητικό σήμα x_0 επί T βήματα ($T = 1000$), έως ότου καταλήξει σε καθαρό ισότροπο θόρυβο $x_T \sim N(0, I)$. Στην **Αντίστροφη Διαδικασία Αποθορυβοποίησης (Reverse Denoising Process)**, ένα νευρωνικό δίκτυο $\epsilon_\theta(x_t, t, c)$, καθοδηγούμενο από τη γλωσσική προτροπή c (*prompt conditioning*), μαθαίνει να προβλέπει και να αφαιρεί τον θόρυβο βήμα προς βήμα, σμιλεύοντας ένα ολοκαίνουργιο, συνεκτικό μέσο:

$$q(x_t \mid x_{t-1}) = N(x_t ; \sqrt{(1-\beta_t)}\, x_{t-1}, \beta_t I) \quad | \quad L_{simple}(\theta) = E_{t,\, x_0,\, \epsilon}\, [\, || \epsilon - \epsilon_\theta(x_t, t, c) ||_2^2 \,]$$

Χάρη στην ιδιότητα των Γκαουσιανών, η εκπαίδευση μεταβαίνει άμεσα σε οποιοδήποτε βήμα t ($x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{(1-\bar{\alpha}_t)}\, \epsilon$). Κατά την παραγωγή, το **Classifier-Free Guidance (CFG)** ενισχύει την πιστότητα στην προτροπή c μέσω $\hat{\epsilon} = \epsilon_\theta(x_t, t, \emptyset) + w \cdot (\epsilon_\theta(x_t, t, c) - \epsilon_\theta(x_t, t, \emptyset))$.

- **Λανθάνουσα Διάχυση (Latent Diffusion Models, LDM):** Αντί να αποθορυβοποιούν εκατομμύρια εικονοστοιχεία RGB υψηλής ανάλυσης, τα σύγχρονα συστήματα συμπιέζουν πρώτα την εικόνα ή το βίντεο μέσω ενός *Variational Autoencoder (VAE)* σε έναν 8× μικρότερο λανθάνοντα χώρο, μειώνοντας κατά τάξεις μεγέθους το υπολογιστικό κόστος.
- **Diffusion Transformers (DiT) & Flow Matching:** Τα μετωπικά μοντέλα εικόνας, βίντεο και ήχου αντικατέστησαν τα συνελικτικά δίκτυα U-Net με επεκτάσιμους **Diffusion Transformers (DiT)** πάνω σε χωροχρονικά τεμάχια (*patches*) και ευθείες τροχιές *Rectified Flow Matching*, επιτυγχάνοντας εντυπωσιακή ποιότητα σε μόλις 20 έως 30 βήματα.
- **Μηχανική Προτροπών & Δημιουργική Ποικιλομορφία:** Όπως αναδείχθηκε ήδη στα Σχήματα 1.1α και 1.1β της σαβάνας στο Κεφάλαιο 1, μία σαφής αρχιτεκτονική προτροπή (υποκείμενο, ατμόσφαιρα, φωτισμός, μετασχηματισμός) καθοδηγεί σφιχτά τη διασταυρούμενη προσοχή, ενώ κάθε διαφορετικός αρχικός σπόρος θορύβου x_T αναδεικνύει τη δημιουργική ποικιλομορφία (*generative diversity*) του λανθάνοντος χώρου.

15.2 Συμβολική Σύνθεση Μουσικής MIDI & Πολυτροπικό Καλειδοσκόπιο (Εργαστήριο 21)

Παράλληλα με τα μοντέλα διάχυσης ήχου σε κυματομορφή, η **Συμβολική Σύνθεση Μουσικής** λειτουργεί στον διακριτό χώρο γεγονότων **MIDI** (*Musical Instrument Digital Interface*), όπου κάθε νότα περιγράφεται ακριβώς από τον αριθμό τονικού ύψους (0 έως 127, με το μεσαίο Do C4 = 60), την ένταση (*velocity*) και τη χρονική διάρκεια σε *ticks*. Αυτή η καθαρή συμβολική αναπαράσταση επιτρέπει άμεση μουσικολογική αντιστοίχιση συναισθήματος σε αρμονία (*Sentiment-to-Harmony Mapping*):

- **Φωτεινή / Θετική Πολικότητα (4 έως 5 αστέρια BERT):** Ενεργοποιεί **Μείζονες συγχορδίες** (π.χ. C, G, Am, F), γρήγορο ζωηρό τέμπο (~**120 BPM**, βήμα 320 ticks), ανοδικές πεντατονικές μελωδίες και ζεστή χρυσαυγή/κυανή χρωματική παλέτα.
- **Στοχαστική / Μελαγχολική Πολικότητα (1 έως 3 αστέρια BERT):** Μεταβαίνει σε **Ελάσσονες λυρικές κλίμακες** (π.χ. Am, F, Dm, E), βραδύτερο εκφραστικό τέμπο (~**80 BPM**, βήμα 460 ticks) και βαθιά κυανό/ιώδη χρωματική παλέτα στον γεωμετρικό σχεδιασμό 8-πλής περιστροφικής συμμετρίας του καλειδοσκοπίου σε **Pygame**.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

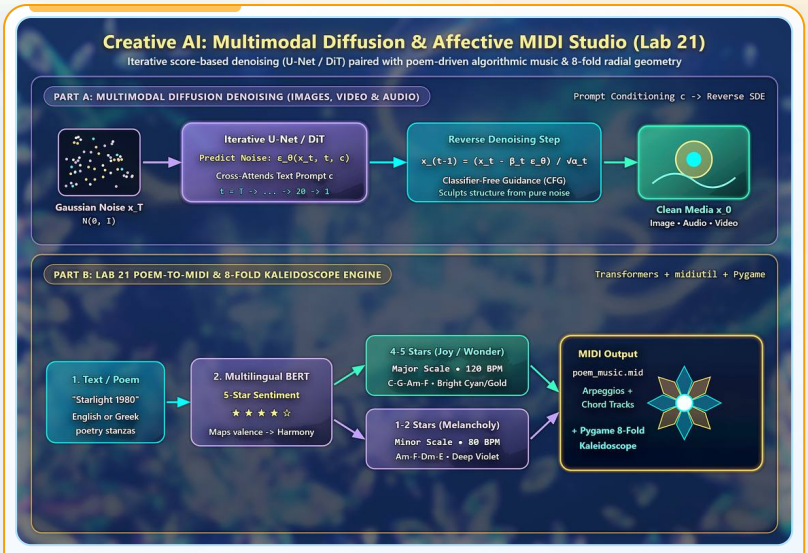
Ένα Βήμα Διάχυσης σε 1 Pixel & Αντιστοίχιση Συναισθήματος (2/5 vs 5/5) σε Νότες MIDI

Μέρος Α (Εμπρόσθια Προσθήκη & Ανάκτηση Θορύβου Διάχυσης): Έστω ένα κανονικοποιημένο pixel $x_0 = 0.80$ σε χρονικό βήμα με συντελεστή διατήρησης σήματος $\bar{\alpha}_t = 0.64$ και τυχαίο δείγμα Γκαουσιανού θορύβου $\epsilon = -0.50$:

1. **Πλάτη Σήματος & Θορύβου:** $\sqrt{\bar{\alpha}_t} = \sqrt{0.64} = \mathbf{0.80}$ (80% αρχικό σήμα) και $\sqrt{(1-\bar{\alpha}_t)} = \sqrt{0.36} = \mathbf{0.60}$ (60% θόρυβος).
2. **Θορυβώδες Pixel x_t:** $x_t = (0.80 \times 0.80) + (0.60 \times (-0.50)) = 0.64 - 0.30 = \mathbf{0.34}$. Εάν το νευρωνικό δίκτυο προβλέψει σωστά $\hat{\epsilon} = -0.50$, αντιστρέφοντας τον τύπο ανακτά ακριβώς $\hat{x}_0 = (0.34 - 0.60 \times (-0.50)) / 0.80 = 0.64 / 0.80 = \mathbf{0.80}$!

Μέρος Β (Από το Συναίσθημα BERT στις Νότες MIDI του Εργαστηρίου 21):

- **Στοχαστικό κείμενο (`2/5 αστέρια`):** Επιλέγεται **Ελάσσονα κλίμακα** (`80 BPM`, βήμα `460 ticks`) με βασική συγχορδία **Am**: `A3 = 57`, `C4 = 60`, `E4 = 64` και βαθιά κυανή/ιώδη παλέτα.
- **Φωτεινό κείμενο (`5/5 αστέρια`):** Επιλέγεται **Μείζονα κλίμακα** (`120 BPM`, βήμα `320 ticks`) με βασική συγχορδία **C**: `C4 = 60`, `E4 = 64`, `G4 = 67` και ζεστή χρυσαυγή παλέτα.



Σχήμα 15.1

Δημιουργική AI: επαναληπτική αποθορυβοποίηση διάχυσης (U-Net / DiT με Classifier-Free Guidance) και πλήρης πολυτροπικός αγωγός του Εργαστηρίου 21 (πολυγλωσσικό BERT 5 αστέρων, αρμονική σύνθεση MIDI 120 BPM Major έναντι 80 BPM Minor, ελληνική αφήγηση gTTS και καλειδοσκόπιο 8-πλής συμμετρίας).

ΕΡΓΑΣΤΗΡΙΟ 21

Πολυγλωσσική Ανάλυση Συναισθήματος BERT, Αλγοριθμική Σύνθεση MIDI, gTTS & Καλειδοσκόπιο Pygame

`code/21_sentiment_music_kaleidoscope.py`

Ενώνει τέσσερα πολυτροπικά υποσυστήματα πάνω σε ελληνικό λογοτεχνικό κείμενο (προεπιλογή: «Αστόφως 1980»): ταξινομεί τη συναισθηματική πολικότητα με `nlptown/bert-base-multilingual-uncased-sentiment` (1 έως 5 αστέρια), συνθέτει πρωτότυπη πολυφωνική μουσική `creative_ai_melody.mid` σε μείζονα ή ελάσσονα κλίμακα μέσω `mido`, δημιουργεί ελληνική φωνητική αφήγηση (`gTTS`) και προβάλλει συγχρονισμένο γεωμετρικό καλειδοσκόπιο οκταπλής συμμετρίας σε 60 FPS (`pygame`), με ασφαλή υποβάθμιση εάν λείπουν προαιρετικές βιβλιοθήκες.

Προτεινόμενος Πειραματισμός: Εισαγάγετε δύο αντίθετα ελληνικά κείμενα (ένα ενθουσιώδες και ένα στοχαστικό) και συγκρίνετε την παραγόμενη αρμονική ακολουθία MIDI (C-G-Am-F έναντι Am-F-Dm-E), το τέμπο και τη χρωματική παλέτα του καλειδοσκοπίου.

PYTHON 3.12+ `code/21_sentiment_music_kaleidoscope.py` Ανάλυση συναισθήματος Transformers, σύνθεση MIDI & καλειδοσκόπιο

```python
import math
from pathlib import Path
import random

# Προαιρετικές εξαρτήσεις πολυτροπικότητας με ασφαλή υποβάθμιση εάν δεν είναι εγκατεστημένες
try:
    from transformers import pipeline as hf_pipeline
except ImportError:
    hf_pipeline = None

try:
    from mido import Message, MidiFile, MidiTrack
except ImportError:
    MidiFile = None

try:
    from gtts import gTTS
except ImportError:
    gTTS = None
```

```
try:
    import pygame
except ImportError:
    pygame = None


# === 1. Προεπιλεγμένο Λογοτεχνικό Κείμενο Εισόδου ===

ASTOFOS_1980_EXCERPT = (
    "Ουράνιο άστρο μες το κύμα του βοριά, τι φως που ξεπροβάλλεις. "
    "Μοναχικό παρέμεινες εδώ στην άγρια νύχτα, τα βλέμματα των ανθρώπων λιγόστεψαν. "
    "Άστρο να, κοροϊδεύω τον αέρα μες την άγρια νύχτα και μένω εδώ. "
    "Το φως σου δεν παύω να βλέπω, παντού ξεφύτρωσαν καθρέφτες στη Γη."
)


# === 2. Πολυγλωσσική Ανάλυση Συναισθήματος (BERT Transformer) ===

def analyze_sentiment_stars(text: str) -> tuple[int, float]:
    """Επιστρέφει βαθμολογία συναισθήματος (1 έως 5 αστέρια) και πιθανότητα εμπιστοσύνης."""
    if hf_pipeline is None:
        print("Η βιβλιοθήκη transformers δεν βρέθηκε. Χρήση προεπιλεγμένου στοχαστικού προφίλ (3
αστέρια).")
        return 3, 1.0

# ... (πλήρης εκτελέσιμος κώδικας στο αρχείο code/21_sentiment_music_kaleidoscope.py του code.zip)
```

15.3 Σε Ποιο Επίπεδο Βρίσκεται η Κατανόηση της Γλώσσας; Το Πείραμα «Αστόφως 1980»

Οι στίχοι που ενσωματώνονται ως προεπιλεγμένο κείμενο στο Εργαστήριο 21 προέρχονται από ανέκδοτη ποιητική συλλογή του συγγραφέα με τίτλο **«Αστόφως 1980»**, γραμμένη το 1980 σε ηλικία 16 ετών. Το ολοκληρωμένο ποίημα περιγράφει ένα δυστοπικό μέλλον όπου η Γη δοκιμάζεται από την ανεξέλεγκτη ανθρώπινη παρέμβαση στο φυσικό περιβάλλον και την επικράτηση της τεχνολογίας έναντι της ανθρώπινης ηθικής.

Επιλέχθηκε σκόπιμα το παρακάτω πυκνό και συμβολικό απόσπασμα, ακριβώς επειδή παρουσιάζει σημαντική ερμηνευτική δυσκολία και ο ίδιος ο δημιουργός του γνωρίζει την αληθινή του πρόθεση. Οι στίχοι δόθηκαν στην ελληνική γλώσσα ως είσοδος σε μεγάλο γλωσσικό μοντέλο αρχιτεκτονικής **Transformers (GPT)** με το ερώτημα να αποκωδικοποιήσει το βαθύτερο νόημά τους.



Σχήμα 15.2

Εικαστική απόδοση του ποιήματος «Αστόφως 1980»: το μοναχικό ουράνιο άστρο μέσα στην άγρια νύχτα.

«Αστόφως 1980» (Απόσπασμα) | Ευάγγελος Β. Πεφάνης

Ουράνιο άστρο μες το κύμα του βοριά,
τι φως που ξεπροβάλλεις.

Μοναχικό παρέμεινες εδώ στην άγρια νύχτα,
τα βλέμματα των ανθρώπων λιγόστεψαν.
Στράφηκαν στα βάθη και χάθηκαν.

Άστρο για μια στιγμή ρίξε την λάμψη σου,
στα χέρια μας που βάλθηκαν τον κόσμο να κυριεύσουν
και ντράπηκαν την θλίψη τους να φανερώσουν.

Στο σπίτι μας που ανεμοσκορπίστηκε στους ουρανούς
και χτύπησε στα κανάλια της απανωτής δυστυχίας.

Σε μας που δεν ψάχνουμε για νίκες, στους έρημους δρόμους
που έλιωσαν κάτω στα πόδια μας, τα βαλμένα να στεριώσουν
στο κρύο γαλάκτωμα του μυαλού μας, σαν έλιωσε μέσα στο κεφάλι μας
και μας τύφλωσε για πάντα.

Άστρο να! κοροϊδεύω τον αέρα μες την άγρια νύχτα και μένω εδώ.
Το φως σου δεν παύω να βλέπω, γύρω ποια δεν γυρνώ.
Παντού ξεφύτρωσαν καθρέφτες στη Γη.

15.4 Η Ενοποιημένη Ερμηνεία του Μοντέλου Transformers

Το μοντέλο ανέλυσε τους στίχους συστηματικά σε έξι εννοιολογικούς άξονες, προσεγγίζοντας με εντυπωσιακή πιστότητα τους συμβολισμούς του ποιητή:

1. **Το «ουράνιο άστρο μες το κύμα του βοριά»:** Πηγή φωτός, αλήθειας και ελπίδας μέσα σε μια σκοτεινή και ταραχώδη νύχτα, σύμβολο σταθερού προσανατολισμού για τον άνθρωπο.
2. **Η μοναξιά του άστρου και τα «βλέμματα που χάθηκαν»:** Εκφράζουν την απομάκρυνση της ανθρωπότητας από τα ανώτερα ιδανικά και την απώλεια πνευματικής καθοδήγησης.
3. **«Στα χέρια μας που βάλθηκαν τον κόσμο να κυριεύσουν»:** Αναδεικνύεται η ανθρώπινη υπεροψία και η δίψα κυριαρχίας, σε αντίθεση με τον φόβο αποδοχής της εσωτερικής ευαισθησίας και θλίψης.
4. **Το «ανεμοσκορπισμένο σπίτι»:** Συμβολίζει την απώλεια σταθερότητας, αρμονίας, αξιών και δεσμών που κρατούν ακέραιη την ανθρώπινη κοινότητα και τον πλανήτη.
5. **Οι «έρημοι δρόμοι» και το «κρύο γαλάκτωμα του μυαλού»:** Αποτυπώνουν την αποξένωση, τη συναισθηματική ψυχρότητα και την τύφλωση που εμποδίζει την αναγνώριση της αλήθειας.
6. **Οι «καθρέφτες που ξεφύτρωσαν παντού στη Γη»:** Σύμβολο αυτογνωσίας, εσωτερικής εξέτασης και αλληλεπίδρασης μεταξύ εσωτερικού και εξωτερικού κόσμου, με τον ποιητή να παραμένει ανυποχώρητα στραμμένος προς το φως του άστρου.

> ★ **Συμπέρασμα του Συγγραφέα από το Πείραμα**
>
> Οι απαντήσεις του μοντέλου ανταποκρίθηκαν πλήρως στο βαθύτερο, ισχυρό και αισιόδοξο μήνυμα που ήθελα να εκφράζει το ποίημα όταν το έγραψα στο μακρινό 1980. Τα συναισθήματα που μου δημιουργούνται από αυτή την ερμηνεία είναι: **έκπληξη, θαυμασμός, προβληματισμός, αλλά και δέος για τις τεχνολογικές εξελίξεις που έρχονται με υπερταχύτατους πλέον ρυθμούς.**



ΚΕΦΑΛΑΙΟ 16

Ηθική, Διακυβέρνηση και η Επιστημονική Επανάσταση της AI

Μαθηματικά αλγοριθμικής δικαιοσύνης και διαφορικής ιδιωτικότητας, μηχανιστική επεξηγησιμότητα & C2PA, η Ευρωπαϊκή Πράξη AI (EU AI Act / GPAI) και οι 5 επιστημονικοί πυλώνες ανακαλύψεων.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Μαθηματικά Δικαιοσύνης & Ιδιωτικότητας:** Αυστηρός ορισμός της Δημογραφικής Ισοτιμίας, της Ισοτιμίας Πιθανοτήτων Σφάλματος (*Equalized Odds*) και της εγγύησης ϵ-Διαφορικής Ιδιωτικότητας.
- **Διαφάνεια & Αυθεντικότητα:** Σύγχρονη Μηχανιστική Επεξηγησιμότητα (Sparse Autoencoders, κυκλώματα ενεργοποίησης) και κρυπτογραφική προέλευση μέσων **C2PA**.
- **Ευρωπαϊκή Πράξη AI (Κανονισμός (ΕΕ) 2024/1689):** Πλήρης κατανόηση της πυραμίδας 4 επιπέδων κινδύνου και των εξειδικευμένων υποχρεώσεων για μοντέλα γενικού σκοπού (**GPAI**).
- **AI για την Επιστήμη:** Οι πέντε καταλυτικοί πυλώνες ανακάλυψης (AlphaFold 3, GraphCast/GenCast, Πλάσμα Σύντηξης, GNoME Υλικά, SKA/Rubin Αστρονομία).

16.1 Μαθηματικά Δικαιοσύνης, Διαφορική Ιδιωτικότητα, Επεξηγησιμότητα & C2PA

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Πώς ένας Αμερόληπτος Προγραμματιστής Μπορεί να Φτιάξει ένα Μεροληπτικό Μοντέλο;

Ένας φοιτητής που ξεκινά τώρα τη Μηχανική Μάθηση συχνά αναρωτιέται: *«Αν γράψω απόλυτα ουδέτερο κώδικα και σβήσω ακόμη και τις στήλες φύλου ή καταγωγής, δεν θα είναι ο αλγόριθμος αυτόματα 100% δίκαιος;»* Η απάντηση είναι **όχι**! Ένα νευρωνικό δίκτυο λειτουργεί σαν **καθρέφτης των ιστορικών δεδομένων**: εάν παλαιότερες ανθρώπινες αποφάσεις περιείχαν ανισότητες (ή εάν αθώες στήλες, όπως ο ταχυδρομικός κώδικας ή το ωράριο, λειτουργούν ως έμμεσοι δείκτες μιας ομάδας), το μοντέλο μαθαίνει να αναπαράγει και να πολλαπλασιάζει τη μεροληψία χωρίς κανείς να την έχει προγραμματίσει σκόπιμα!

- **TPR & FPR με απλά λόγια:** Το **TPR (True Positive Rate)** μετρά πόσους από τους πραγματικά ικανούς υποψηφίους εγκρίνει σωστά το σύστημα, ενώ το **FPR (False Positive Rate)** πόσους μη επιλέξιμους εγκρίνει κατά λάθος.
- **Διαφορική Ιδιωτικότητα (ϵ):** Μια μαθηματική «ασπίδα ελεγχόμενου θορύβου» που εγγυάται ότι κανείς δεν μπορεί να ξεχωρίσει αν το δικό σας αρχείο υπήρχε στα δεδομένα εκπαίδευσης.
- **C2PA & EU AI Act:** Το **C2PA** είναι μια κρυπτογραφική «σφραγίδα γνησιότητας» για εικόνες και βίντεο, ενώ το **EU AI Act** είναι ο πρώτος ευρωπαϊκός κανονισμός που ταξινομεί κάθε σύστημα AI ανάλογα με τον κίνδυνο για τους πολίτες.

Καθώς τα νευρωνικά συστήματα ενσωματώνονται στην ιατρική, τη δικαιοσύνη, την εκπαίδευση, την ενέργεια και τις μεταφορές, η τεχνική αρτιότητα δεν αρκεί: απαιτείται **μαθηματικά επαληθεύσιμη ηθική μηχανική**. Οι πέντε κεντρικοί άξονες αξιοπιστίας που οφείλει να ελέγχει κάθε μηχανικός και ερευνητής AI περιλαμβάνουν:

- **Αλγοριθμική Μεροληψία & Τυπική Δικαιοσύνη (Fairness):** Εάν τα ιστορικά δεδομένα εκπαίδευσης περιέχουν κοινωνικές ανισότητες, το μοντέλο θα τις αναπαράγει σε μαζική κλίμακα. Δύο θεμελιώδη μαθηματικά κριτήρια ως προς ένα προστατευόμενο χαρακτηριστικό $A \in \{0, 1\}$ είναι η **Δημογραφική Ισοτιμία (Demographic Parity)**, που απαιτεί ίσο ποσοστό θετικών προβλέψεων $P(\hat{Y} = 1 \mid A = 0) = P(\hat{Y} = 1 \mid A = 1)$, και η αυστηρότερη **Ισοτιμία Πιθανοτήτων Σφάλματος (Equalized Odds)**, που απαιτεί ίσα ποσοστά αληθώς και ψευδώς θετικών προβλέψεων (*TPR* και *FPR*) ανά ομάδα. Τα θεωρήματα αδυναμίας (*Chouldechova / Kleinberg*) αποδεικνύουν ότι όταν οι βασικοί ρυθμοί διαφέρουν μεταξύ ομάδων, δεν ικανοποιούνται ταυτόχρονα όλα τα κριτήρια δικαιοσύνης, καθιστώντας τη στάθμιση συνειδητή κοινωνική και επιστημονική απόφαση.

- **Διαφορική Ιδιωτικότητα (ϵ-Differential Privacy, Dwork et al.):** Η απλή «ανωνυμοποίηση» πινάκων αποτυγχάνει απέναντι σε επιθέσεις επαναταυτοποίησης και εξαγωγής δεδομένων εκπαίδευσης. Ένας τυχαιοποιημένος αλγόριθμος M (π.χ. εκπαίδευση DP-SGD με αποκοπή κλίσεων και βαθμονομημένο Γκαουσιανό θόρυβο) παρέχει αυστηρή μαθηματική εγγύηση ότι η συμμετοχή ή μη ενός μεμονωμένου ατόμου στο σύνολο δεδομένων μεταβάλλει ανεπαίσθητα την κατανομή εξόδου.
- **Μηχανιστική Επεξηγησιμότητα (Mechanistic Interpretability):** Πέρα από τις μεταγενέστερες μεθόδους απόδοσης χαρακτηριστικών (SHAP, Grad-CAM), η σύγχρονη έρευνα χαρτογραφεί απευθείας τα εσωτερικά κυκλώματα των Transformers μέσω *Sparse Autoencoders (SAEs)* και *activation patching*, αποσυνθέτοντας τους υπέρθετους νευρώνες σε ανθρωπίνως ερμηνεύσιμες έννοιες και ελέγχοντας μηχανιστικά τυχόν παραπλανητικές συμπεριφορές.
- **Ψευδαισθήσεις & Κρυπτογραφική Προέλευση Περιεχομένου (C2PA):** Καθώς τα παραγωγικά μοντέλα εικόνας, φωνής και βίντεο γίνονται φωτορεαλιστικά, το ανοιχτό πρότυπο **C2PA (Coalition for Content Provenance and Authenticity)** ενσωματώνει κρυπτογραφικά υπογεγραμμένα μεταδεδομένα προέλευσης και ιστορικού επεξεργασίας (*Content Credentials*), προστατεύοντας τη δημοσιογραφία, την επιστημονική ακεραιότητα και τον δημόσιο διάλογο.
- **Ενεργειακό & Περιβαλλοντικό Αποτύπωμα:** Βελτιστοποίηση αρχιτεκτονικών, κβαντοποίηση 4-bit (Κεφάλαιο 10), επιμελής επαναχρησιμοποίηση προσαρμογέων LoRA (Κεφάλαιο 11) και λειτουργία κέντρων δεδομένων με ανανεώσιμες πηγές ενέργειας.

$$\text{Equalized Odds: } P(\hat{Y} = 1 \mid Y = y, A = 0) = P(\hat{Y} = 1 \mid Y = y, A = 1), \quad y \in \{0, 1\}$$

$$\epsilon\text{-Differential Privacy: } P[\, M(D) \in S \,] \leq \exp(\epsilon) \cdot P[\, M(D') \in S \,]$$

Για κάθε ζεύγος γειτονικών συνόλων δεδομένων D και D' που διαφέρουν σε ακριβώς μία εγγραφή ατόμου και για κάθε σύνολο αποτελεσμάτων S: όσο μικρότερος ο προϋπολογισμός ιδιωτικότητας $\epsilon > 0$, τόσο ισχυρότερη η μαθηματική προστασία έναντι διαρροής προσωπικών δεδομένων.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Έλεγχος Δικαιοσύνης (Demographic Parity vs Equalized Odds) & Όριο Ιδιωτικότητας ε = 0.5

Μέρος Α (Αριθμητικός Έλεγχος Δικαιοσύνης σε 200 Αιτήσεις Υποτροφίας): Έστω δύο ομάδες 100 υποψηφίων η καθεμία (**Ομάδα Α** και **Ομάδα Β**), όπου και στις δύο υπάρχουν ακριβώς 50 πραγματικά επιλέξιμοι ($Y = 1$) και 50 μη επιλέξιμοι ($Y = 0$):

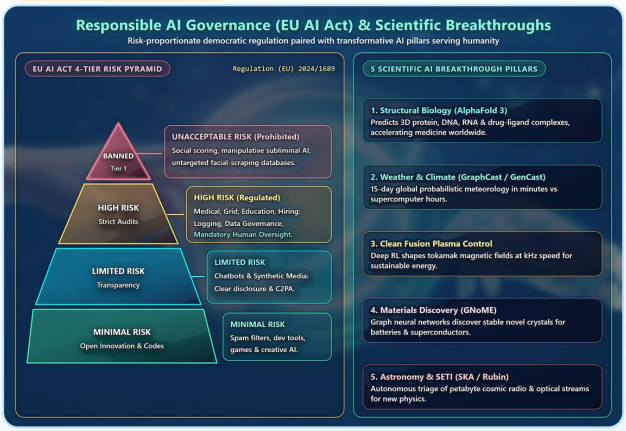
1. **Ομάδα Α:** Το μοντέλο εγκρίνει 44/50 επιλέξιμους ($\mathbf{TPR_A = 88\%}$) και 6/50 μη επιλέξιμους ($\mathbf{FPR_A = 12\%}$). Συνολικές εγκρίσεις: $(44 + 6)/100 = \mathbf{50\%}$.
2. **Ομάδα Β:** Το μοντέλο εγκρίνει 38/50 επιλέξιμους ($\mathbf{TPR_B = 76\%}$) και 12/50 μη επιλέξιμους ($\mathbf{FPR_B = 24\%}$). Συνολικές εγκρίσεις: $(38 + 12)/100 = \mathbf{50\%}$.
3. **Διάγνωση:** Με βάση μόνο τη **Δημογραφική Ισοτιμία**, και οι δύο ομάδες έχουν *50%* έγκριση (διαφορά 0%, φαινομενικά δίκαιο!). Όμως ο έλεγχος **Equalized Odds** αποκαλύπτει σοβαρή απόκλιση **12 ποσοστιαίων μονάδων** στο TPR ($88\% - 76\% = 12\%$) και 12 μονάδων στο FPR ($24\% - 12\% = 12\%$): ικανοί υποψήφιοι της Ομάδας Β αδικούνται συστηματικά, οπότε το μοντέλο απορρίπτεται!

Μέρος Β (Τι Σημαίνει ϵ = 0.5 στη Διαφορική Ιδιωτικότητα): Υπολογίζουμε $exp(0.5) = e^{0.5} \approx \mathbf{1.65}\times$. Άρα η προσθήκη ή αφαίρεση ενός μεμονωμένου πολίτη από τη βάση δεδομένων μπορεί να μεταβάλει την πιθανότητα οποιουδήποτε αποτελέσματος το πολύ κατά **1.65 φορές** (ισχυρή προστασία, σε αντίθεση με ένα χαλαρό $\epsilon = 5.0$ όπου $e^5 \approx 148.4\times$).

16.2 Η Ευρωπαϊκή Πράξη για την Τεχνητή Νοημοσύνη (EU AI Act, Κανονισμός 2024/1689)

Η Ευρωπαϊκή Ένωση θέσπισε με τον **Κανονισμό (ΕΕ) 2024/1689 (EU AI Act)** το πρώτο ολοκληρωμένο νομικό πλαίσιο παγκοσμίως για την Τεχνητή Νοημοσύνη, βασισμένο σε αναλογική προσέγγιση **τεσσάρων επιπέδων κινδύνου** σε συνδυασμό με ειδικό κεφάλαιο για τα **Μοντέλα AI Γενικού Σκοπού (GPAI, General-Purpose AI Models)**:

Επίπεδο Κινδύνου / Κατηγορία	Χαρακτηριστικά Παραδείγματα	Ρυθμιστική Απαίτηση (Κανονισμός 2024/1689)
1. Απαράδεκτος Κίνδυνος	Κοινωνική βαθμολόγηση πολιτών, υποσυνείδητη χειραγώγηση, αδιάκριτη συλλογή προσώπων από το διαδίκτυο	Πλήρης απαγόρευση κυκλοφορίας και χρήσης εντός της Ευρωπαϊκής Ένωσης
2. Υψηλός Κίνδυνος	Ιατροτεχνολογικά συστήματα, κρίσιμες υποδομές (ενέργεια/δίκτυα), επιλογή προσωπικού, εκπαιδευτική αξιολόγηση	Διαχείριση κινδύνου, διακυβέρνηση δεδομένων, καταγραφή συμβάντων (logging), τεκμηρίωση & υποχρεωτική ανθρώπινη εποπτεία
3. Περιορισμένος Κίνδυνος	Συστήματα συνομιλίας (chatbots), συνθετικά μέσα (deepfakes, παραγόμενες εικόνες, ήχος & βίντεο)	Υποχρεωτική διαφάνεια προς τον χρήστη και μηχανικά αναγνώσιμη σήμανση προέλευσης (C2PA / υδατογράφηση)
4. Ελάχιστος Κίνδυνος	Φίλτρα ανεπιθύμητης αλληλογραφίας, εργαλεία προγραμματιστών, βιντεοπαιχνίδια, δημιουργικά εργαλεία	Ελεύθερη ανάπτυξη και καινοτομία με ενθάρρυνση εθελοντικών κωδίκων δεοντολογίας
Μοντέλα Γενικού Σκοπού (GPAI)	Θεμελιώδη LLMs & πολυτροπικά μοντέλα (με επιπλέον καθεστώς συστημικού κινδύνου άνω των 10^{25} FLOPs)	Τεχνική τεκμηρίωση, πολιτική πνευματικών δικαιωμάτων, σύνοψη δεδομένων εκπαίδευσης & κόκκινες δοκιμές (Red-Teaming) για συστημικά μοντέλα (με εξαιρέσεις ανοιχτής έρευνας & open-source χωρίς συστημικό κίνδυνο)



Σχήμα 16.1

Διακυβέρνηση Υπεύθυνης AI και Επιστημονικές Ανακαλύψεις: η πυραμίδα 4 επιπέδων κινδύνου του Κανονισμού (ΕΕ) 2024/1689 (EU AI Act) δίπλα στους 5 καταλυτικούς πυλώνες επιστημονικής προόδου της AI.

16.3 Η Τεχνητή Νοημοσύνη ως Επιταχυντής της Επιστήμης: Οι 5 Πυλώνες Ανακαλύψεων

Πέρα από τις εμπορικές εφαρμογές, η βαθιά μάθηση μετασχηματίζει ριζικά τη θεμελιώδη επιστημονική ανακάλυψη σε παγκόσμια κλίμακα, λειτουργώντας ως ψηφιακός επιταχυντής πέντε στρατηγικών τομέων:

1. Δομική Βιολογία (AlphaFold 3)

Μοντέλα διάχυσης και γεωμετρικής βαθιάς μάθησης προβλέπουν με ατομική ακρίβεια τη τρισδιάστατη δομή πρωτεϊνών, DNA, RNA και φαρμακευτικών συμπλόκων, μειώνοντας κατά έτη την ανακάλυψη θεραπειών.

2. Κλίμα & Καιρός (GraphCast / GenCast)

Γραφο-νευρωνικά και πιθανοτικά μοντέλα διάχυσης παράγουν παγκόσμιες προγνώσεις 15 ημερών σε λίγα λεπτά αντί ωρών υπερυπολογιστή, ενώ το **Edge AI / TinyML** παρακολουθεί περιβαλλοντικούς αισθητήρες με κατανάλωση milliwatt.

3. Καθαρή Πυρηνική Σύντηξη

Η Βαθιά Ενισχυτική Μάθηση ελέγχει σε συχνότητες kilohertz τα μαγνητικά πηνία αντιδραστήρων tokamak, σμιλεύοντας δυναμικά το πλάσμα υψηλής θερμοκρασίας για σταθερή καθαρή ενέργεια.

4. Ανακάλυψη Νέων Υλικών (GNoME)

Δίκτυα Γράφων (GNNs) προβλέπουν τη θερμοδυναμική ευστάθεια εκατομμυρίων υποψήφιων κρυστάλλων, αποκαλύπτοντας νέους στερεούς ηλεκτρολύτες μπαταριών, καταλύτες και υπεραγωγούς.

5. Αστρονομία & SETI (SKA / Vera C. Rubin)

Πολυτροπικά θεμελιώδη μοντέλα και πράκτορες MCP (Κεφάλαιο 14) φιλτράρουν σε πραγματικό χρόνο petabytes οπτικών και ραδιοαστρονομικών ροών για την ανακάλυψη νέας φυσικής, FRBs και τεχνοϋπογραφών.

16.4 Πλαίσιο Μηχανικής Επαλήθευσης Υπεύθυνης AI στην Πράξη

Πριν από την ένταξη οποιουδήποτε μοντέλου μηχανικής μάθησης σε πραγματική υπηρεσία, οι επιστημονικές και μηχανικές ομάδες εφαρμόζουν πέντε συστηματικούς ελέγχους αξιοπιστίας:

Άξονας Ελέγχου	Κρίσιμο Ερώτημα Επαλήθευσης	Πρακτικό Μέτρο Διασφάλισης
Ιχνηλασιμότητα Δεδομένων	Ποια είναι η προέλευση, η άδεια χρήσης και η αντιπροσωπευτικότητα του συνόλου;	Κάρτες Δεδομένων (Datasheets) & έλεγχος ανισορροπίας ομάδων
Ισότητα Σφαλμάτων	Παρουσιάζει το μοντέλο υψηλότερο ποσοστό σφάλματος σε συγκεκριμένες υποομάδες;	Αξιολόγηση Equalized Odds (TPR/FPR) ξεχωριστά ανά δημογραφικό ή φυσικό τομέα
Επεξηγησιμότητα	Μπορεί ένας ειδικός να κατανοήσει ποια χαρακτηριστικά οδήγησαν στην απόφαση;	Χάρτες σημαντικότητας (Grad-CAM, SHAP), μηχανιστικοί έλεγχοι & Model Cards
Ανθεκτικότητα & Ασφάλεια	Πώς αντιδρά το σύστημα σε θορυβώδεις, εκτός κατανομής ή κακόβουλες εισόδους;	Δοκιμές αντοχής (Red-Teaming), όρια εμπιστοσύνης & υπογραφές C2PA
Ανθρώπινη Εποπτεία	Διατηρεί ο άνθρωπος την τελική ευθύνη και δυνατότητα παρέμβασης;	Μηχανισμοί Human-in-the-Loop για κάθε απόφαση υψηλού αντικτύπου

★ **Κεντρική Αρχή Ηθικής Μηχανικής**

Η ρύθμιση και η ηθική δεν αποτελούν τροχοπέδη στην καινοτομία, αλλά το θεμέλιο της κοινωνικής εμπιστοσύνης: μόνο αξιόπιστα, διαφανή και επαληθεύσιμα συστήματα Τεχνητής Νοημοσύνης μπορούν να ενταχθούν με ασφάλεια στην ιατρική, την εκπαίδευση και τις κρίσιμες υποδομές.

ΚΕΦΑΛΑΙΟ 17

Κβαντικοί Υπολογιστές, Κβαντική AI & Μετακβαντική Κρυπτογραφία

Γεωμετρία Σφαίρας Bloch, θεμελιώδεις κβαντικές πύλες (H, X, Y, Z, CNOT), διεμπλοκή Bell, αλγόριθμοι Shor/Grover, πρότυπα NIST PQC (FIPS 203/204/205) και υβριδικά κυκλώματα VQC.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Γεωμετρία Qubit & Σφαίρα Bloch:** Ακριβής αναπαράσταση της υπέρθεσης $|\psi\rangle$ με τις γωνίες θ και φ και υπολογισμός πιθανοτήτων μέτρησης κατά τον Κανόνα Born.
- **Μοναδιαίες Πύλες & Διεμπλοκή Bell:** Μαθηματική δράση των πυλών *H, X, Y, Z, CNOT* και παραγωγή της μέγιστα διεμπλεγμένης κατάστασης Bell $(|00\rangle + |11\rangle)/\sqrt{2}$.
- **Κβαντική Πολυπλοκότητα & NIST PQC:** Ανάλυση της κρυπτογραφικής απειλής του Shor ($O((\log N)^3)$) έναντι του Grover ($O(\sqrt{N})$) και των νέων προτύπων **FIPS 203 (ML-KEM)**, **FIPS 204 (ML-DSA)** και **FIPS 205 (SLH-DSA)**.
- **Λογικά Qubits & Υβριδική Κβαντική ML (VQC):** Κατανόηση της Κβαντικής Διόρθωσης Σφαλμάτων (Surface Codes) και της εκπαίδευσης παραμετρικών κυκλωμάτων $U(\theta)$ με τον κανόνα μετατόπισης παραμέτρου.

17.1 Από το Κλασικό Bit στο Κβαντικό Qubit & η Σφαίρα Bloch

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Το Νόμισμα, το Γυροσκόπιο & τα Συσχετισμένα Ζάρια: Κβαντική Υπολογιστική χωρίς Μυστήριο

Ένα κλασικό **bit** μοιάζει με ένα **νόμισμα ξαπλωμένο επίπεδα στο τραπέζι**: δείχνει αυστηρά είτε `0` (κορώνα) είτε `1` (γράμματα). Αντίθετα, ένα **qubit** μοιάζει με ένα **περιστρεφόμενο γυροσκόπιο στο κέντρο μιας υδρογείου σφαίρας (Σφαίρα Bloch)**: πριν το σταματήσουμε για μέτρηση, ο άξονάς του μπορεί να δείχνει προς οποιοδήποτε γεωγραφικό πλάτος (θ, που καθορίζει πόσο κοντά βρίσκεται στον Βόρειο Πόλο $|0\rangle$ ή στον Νότιο Πόλο $|1\rangle$) και γεωγραφικό μήκος (φ, τη σχετική κβαντική φάση). Όσο για την **Κβαντική Διεμπλοκή (Entanglement)**, φανταστείτε ένα ζεύγος συσχετισμένων «μαγικών ζαριών»: όποτε ρίξετε το πρώτο στην Πάτρα και δείξει 0, το ταίρι του στην Αθήνα θα δείξει εγγυημένα 0, **χωρίς όμως καμία επικοινωνία ταχύτερη από το φως**, διότι κάθε παρατηρητής βλέπει από μόνος του απόλυτα τυχαία 50/50 αποτελέσματα μέχρι να συγκρίνουν τις σημειώσεις τους μέσω κλασικού δικτύου!

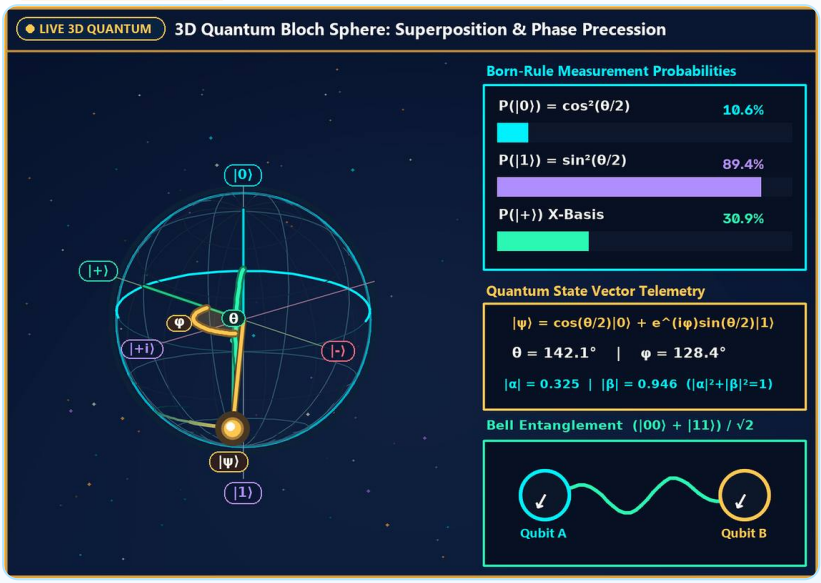
- **Σημειογραφία Dirac ($|0\rangle$, $|1\rangle$):** Ο καθιερωμένος μαθηματικός τρόπος γραφής των κβαντικών διανυσμάτων κατάστασης.
- **Κανόνας Born & NISQ:** Ο Κανόνας Born μετατρέπει τα κβαντικά πλάτη σε πιθανότητες μέτρησης, ενώ **NISQ** ονομάζονται οι σημερινοί θορυβώδεις κβαντικοί επεξεργαστές μεσαίας κλίμακας.
- **Μετακβαντική Κρυπτογραφία (PQC):** Σύγχρονοι κρυπτογραφικοί αλγόριθμοι που τρέχουν σε απλούς κλασικούς υπολογιστές αλλά αντέχουν απέναντι σε μελλοντικούς κβαντικούς.

Ένας κλασικός υπολογιστής επεξεργάζεται δυαδικά ψηφία (*bits*) που βρίσκονται κάθε στιγμή αποκλειστικά στην κατάσταση `0` ή `1`. Ένας **κβαντικός υπολογιστής** αξιοποιεί τις αρχές της κβαντομηχανικής πάνω σε **κβαντικά ψηφία (qubits)**:

- **Υπέρθεση (Superposition):** Ένα qubit μπορεί να βρίσκεται σε μιγαδικό γραμμικό συνδυασμό των βασικών καταστάσεων $|0\rangle$ και $|1\rangle$ ταυτόχρονα, αναπαριστώμενο γεωμετρικά ως σημείο στην επιφάνεια της **Σφαίρας Bloch**.
- **Κβαντική Διεμπλοκή (Entanglement):** Πολλαπλά qubits συσχετίζονται μη κλασικά έτσι ώστε η κοινή τους κατάσταση να μην παραγοντοποιείται σε επιμέρους καταστάσεις, εκτείνοντας έναν εκθετικά μεγάλο χώρο Hilbert διάστασης 2^n (μόλις 300 διεμπλεγμένα qubits ξεπερνούν τον αριθμό ατόμων στο παρατηρήσιμο Σύμπαν).
- **Κβαντική Συμβολή (Interference):** Οι κβαντικοί αλγόριθμοι ενορχηστρώνουν τις σχετικές φάσεις φ ώστε τα πλάτη πιθανότητας των ορθών απαντήσεων να ενισχύονται εποικοδομητικά και των λανθασμένων να αλληλοαναιρούνται πριν από την τελική μέτρηση.

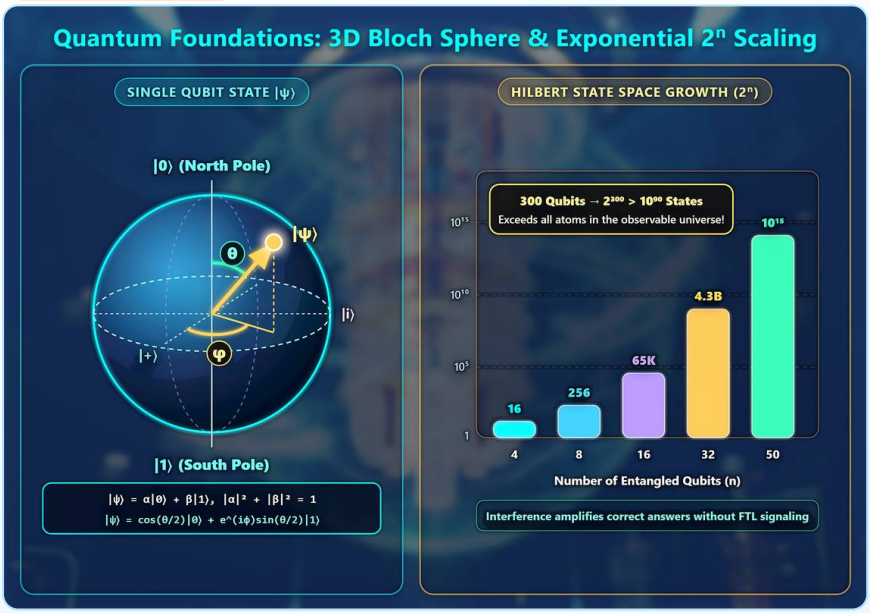
$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \cos(\theta/2)|0\rangle + e^{i\varphi}\sin(\theta/2)|1\rangle \quad | \quad |\alpha|^2 + |\beta|^2 = 1$$

Η πολική γωνία θ ∈ [0, π] καθορίζει τις πιθανότητες μέτρησης (Κανόνας Born) $P(|0\rangle) = |\alpha|^2 = \cos^2(\theta/2)$ και $P(|1\rangle) = |\beta|^2 = \sin^2(\theta/2)$, ενώ η αζιμουθιακή γωνία φ ∈ [0, 2π) ορίζει τη σχετική κβαντική φάση στη Σφαίρα Bloch.



Σχήμα 17.1

Τρισδιάστατη Σφαίρα Bloch σε πραγματικό χρόνο: διάνυσμα κατάστασης |ψ⟩ με πολική γωνία θ και αζιμουθιακή φάση φ, ζωντανές πιθανότητες μέτρησης Κανόνα Born και ζεύγος διεμπλεγμένων qubits Bell.



Σχήμα 17.2

Συνοπτική γεωμετρία της Σφαίρας Bloch με τις γωνίες θ και φ, κβαντική υπέρθεση, εκθετική κλιμάκωση χώρου καταστάσεων 2^n και εποικοδομητική συμβολή.

17.2 Θεμελιώδεις Κβαντικές Πύλες (H, X, Y, Z, CNOT) & Κατάσταση Bell

Μαθηματικά, κάθε κβαντική πύλη είναι ένας **μοναδιαίος πίνακας (Unitary Matrix, $U^\dagger U = I$)** που περιστρέφει το διάνυσμα κατάστασης πάνω στη Σφαίρα Bloch διατηρώντας το άθροισμα πιθανοτήτων ίσο με 1 και καθιστώντας την εξέλιξη πλήρως αντιστρέψιμη πριν από τη μέτρηση. Πέντε στοιχειώδεις πύλες αποτελούν τα δομικά στοιχεία κάθε κβαντικού κυκλώματος:

Κβαντική Πύλη	Μαθηματική Δράση & Γεωμετρία	Ρόλος στα Κβαντικά Κυκλώματα
Hadamard (H)	$\|0\rangle \to (\|0\rangle + \|1\rangle)/\sqrt{2} = \|+\rangle$ | $\|1\rangle \to (\|0\rangle - \|1\rangle)/\sqrt{2} = \|-\rangle$	Δημιουργεί ισόπιθανη κβαντική υπέρθεση από κλασική κατάσταση βάσης
Pauli-X (NOT)	$\|0\rangle \leftrightarrow \|1\rangle$ (περιστροφή 180° γύρω από τον άξονα X)	Κβαντική αντιστροφή bit (*bit-flip*)
Pauli-Y	$\|0\rangle \to i\|1\rangle$, $\|1\rangle \to -i\|0\rangle$ (περιστροφή 180° γύρω από τον άξονα Y)	Ταυτόχρονη αντιστροφή bit και μετατόπιση φάσης
Pauli-Z	$\|0\rangle \to \|0\rangle$, $\|1\rangle \to -\|1\rangle$ (περιστροφή 180° γύρω από τον άξονα Z)	Αντιστροφή σχετικής φάσης (*phase-flip*) κρίσιμη για τη συμβολή
CNOT (2 Qubits)	Εφαρμόζει X στο qubit στόχο *μόνο* όταν το qubit ελέγχου είναι $\|1\rangle$	Η θεμελιώδης πύλη 2 qubits που παράγει **κβαντική διεμπλοκή**

Εφαρμόζοντας απλώς μία πύλη Hadamard στο πρώτο qubit και κατόπιν μία πύλη CNOT στα δύο qubits, παράγουμε τη διάσημη **Κατάσταση Διεμπλοκής Bell ($|\Phi^+\rangle$)**:

$$|\Phi^+\rangle = \text{CNOT} \cdot (\text{H} \otimes \text{I})\,|00\rangle = (\,|00\rangle + |11\rangle\,)\,/\,\sqrt{2}$$

Η κατάσταση $|\Phi^+\rangle$ δεν μπορεί να γραφεί ως γινόμενο $|\psi_A\rangle \otimes |\psi_B\rangle$: μόλις μετρήσουμε το Qubit A (50% «0» ή 50% «1»), η κατάσταση του Qubit B καθορίζεται ακαριαία με 100% συσχέτιση, ανεξάρτητα από τη φυσική απόσταση.

17.3 Κβαντική Πολυπλοκότητα (Shor vs Grover), Πρότυπα NIST PQC & Υβριδικά VQC

Δύο εμβληματικοί κβαντικοί αλγόριθμοι καθορίζουν το πού οι κβαντικοί υπολογιστές προσφέρουν πραγματικό αλγοριθμικό πλεονέκτημα και γιατί απαιτείται άμεση αναβάθμιση της παγκόσμιας κυβερνοασφάλειας:

- **Αλγόριθμος Shor (Εκθετική Επιτάχυνση, *O((log N)³)*):** Αξιοποιώντας τον Κβαντικό Μετασχηματισμό Fourier (QFT) για εύρεση περιόδου, παραγοντοποιεί τεράστιους ακεραίους και επιλύει διακριτούς λογαρίθμους σε πολυωνυμικό χρόνο. Μόλις ωριμάσουν ανεκτικοί σε σφάλματα κβαντικοί υπολογιστές χιλιάδων λογικών qubits, θα καταρρίψουν πλήρως την ασύμμετρη κρυπτογραφία **RSA** και **ECC** που προστατεύει σήμερα το διαδίκτυο (HTTPS/TLS, ψηφιακές υπογραφές, VPN).
- **Αλγόριθμος Grover (Τετραγωνική Επιτάχυνση, *O(√N)*):** Ενισχύει επαναληπτικά το πλάτος της ζητούμενης λύσης σε αδόμητη αναζήτηση *N* στοιχείων σε *O(√N)* βήματα αντί *O(N)*. Υποδιπλασιάζει την αποτελεσματική ασφάλεια των συμμετρικών κλειδιών, οπότε ο διπλασιασμός μήκους (π.χ. **AES-256** και **SHA-384/512**) διατηρεί ισχυρή μετακβαντική αντοχή 128 bits.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Πιθανότητες Κανόνα Born για θ = 60° & Κλασική (2^30) έναντι Κβαντικής Αναζήτησης Grover

Μέρος Α (Υπολογισμός Πιθανοτήτων Μέτρησης στη Σφαίρα Bloch για θ = 60°): Αν ένα qubit προετοιμαστεί σε πολική γωνία `θ = 60°` (οπότε η ημιγωνία είναι `θ/2 = 30°`), εφαρμόζουμε απευθείας τον Κανόνα Born:

1. **Πιθανότητα μέτρησης |0⟩:** $P(|0\rangle) = \cos^2(30°) = (\sqrt{3}/2)^2 = 3/4 =$ ***0.75 (75%)***.
2. **Πιθανότητα μέτρησης |1⟩:** $P(|1\rangle) = \sin^2(30°) = (1/2)^2 = 1/4 =$ ***0.25 (25%)***.
3. **Έλεγχος Κανονικοποίησης:** *75% + 25% =* ***100%***! Σε 1.000 επαναλαμβανόμενες μετρήσεις (*shots*) αναμένουμε ~750 φορές «0» και ~250 φορές «1».

Μέρος Β (Γιατί ο Grover Επιταχύνει Δραματικά την Αδόμητη Αναζήτηση): Έστω μία αταξινόμητη βάση $N = 2^{30} =$ ***1.073.741.824*** (~1.07 δισεκατομμύρια) υποψήφιων κλειδιών. Ένας κλασικός αλγόριθμος χρειάζεται έως $\mathbf{2^{30}}$ ελέγχους (μέσο όρο ~537 εκατομμύρια), ενώ ο κβαντικός αλγόριθμος **Grover** εντοπίζει τη λύση σε τάξη μεγέθους $\sqrt{(2^{30})} = 2^{15} =$ ***32.768 βήματα*** (~32.768× λιγότερες κλήσεις ελέγχου!).

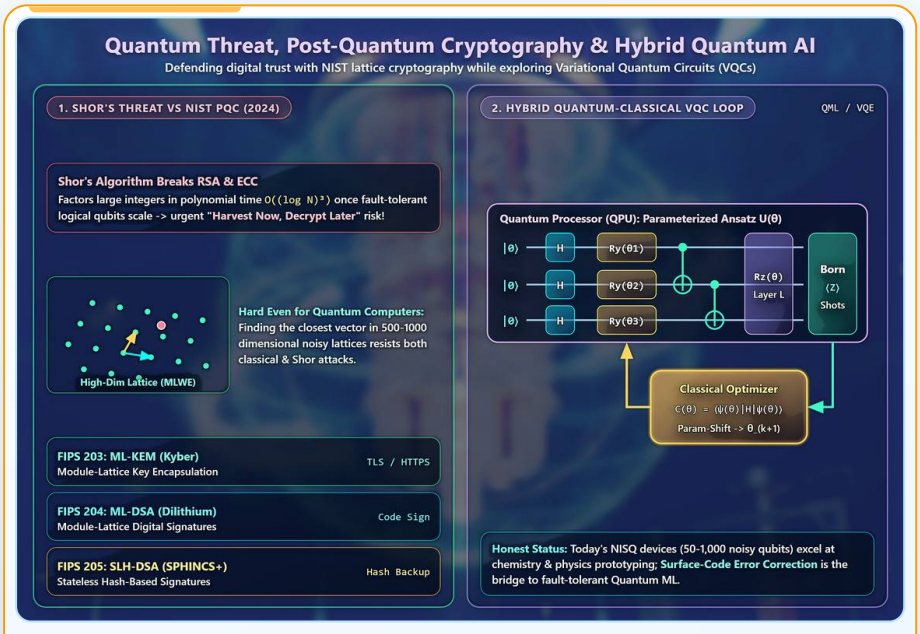
Πρότυπα Μετακβαντικής Κρυπτογραφίας NIST (2024): FIPS 203, FIPS 204 & FIPS 205

Για να αμυνθεί απέναντι στον αλγόριθμο Shor, ο οργανισμός **NIST** οριστικοποίησε τρία νέα παγκόσμια πρότυπα **Μετακβαντικής Κρυπτογραφίας (Post-Quantum Cryptography, PQC)** που εκτελούνται αποδοτικά σε κλασικούς υπολογιστές αλλά αντιστέκονται σε κβαντικές επιθέσεις:

- **FIPS 203 (ML-KEM, πρώην CRYSTALS-Kyber):** Μηχανισμός ενθυλάκωσης κλειδιών (*Key Encapsulation Mechanism*) για ασφαλείς συνδέσεις TLS/HTTPS, βασισμένος στη μαθηματική δυσκολία του προβλήματος **Module Learning With Errors (MLWE)** (εύρεση πλησιέστερου διανύσματος σε θορυβώδη πλέγματα 500 έως 1000 διαστάσεων).
- **FIPS 204 (ML-DSA, πρώην CRYSTALS-Dilithium):** Πρότυπο ψηφιακών υπογραφών βασισμένο σε δομικά πλέγματα για πιστοποιητικά, υπογραφή λογισμικού και ταυτοποίηση.
- **FIPS 205 (SLH-DSA, πρώην SPHINCS+):** Πρότυπο ψηφιακών υπογραφών χωρίς κατάσταση (*stateless*) που στηρίζεται αποκλειστικά σε κρυπτογραφικές συναρτήσεις κατακερματισμού (hash), παρέχοντας ανεξάρτητη μαθηματική εφεδρεία σε περίπτωση μελλοντικής προόδου επί των πλεγμάτων.

Λογικά Qubits Διόρθωσης Σφαλμάτων & Υβριδικά Μεταβλητικά Κυκλώματα (VQC)

Η μεγαλύτερη φυσική πρόκληση των σημερινών συσκευών **NISQ** (50 έως ~1.000 θορυβώδη qubits) είναι η *αποσυνοχή* (η απώλεια της ευαίσθητης κβαντικής κατάστασης λόγω περιβαλλοντικού θορύβου). Πειραματικά ορόσημα στην **Κβαντική Διόρθωση Σφαλμάτων (Surface-Code QEC)** απέδειξαν ότι η ομαδοποίηση πολλών φυσικών qubits σε ένα **Λογικό Qubit** κάτω από το κρίσιμο κατώφλι σφάλματος μειώνει εκθετικά τον ρυθμό λαθών όσο μεγαλώνει η απόσταση του κώδικα. Παράλληλα, στην **Υβριδική Κβαντική Μηχανική Μάθηση (QML)**, κλασικοί βελτιστοποιητές σε GPU εκπαιδεύουν **Μεταβλητικά Κβαντικά Κυκλώματα (Variational Quantum Circuits, VQC)**: ένας παραμετρικός κβαντικός μετασχηματισμός $U(\theta)$ (πύλες περιστροφής $R_y(\theta)$, $R_z(\theta)$ και CNOT) προετοιμάζει την κατάσταση $|\psi(\theta)\rangle$, μετράται η αναμενόμενη τιμή $C(\theta) = \langle\psi(\theta)|H|\psi(\theta)\rangle$ μέσω επαναλαμβανόμενων μετρήσεων (*shots*) και οι κλίσεις υπολογίζονται αναλυτικά στο ίδιο το υλικό μέσω του **Κανόνα Μετατόπισης Παραμέτρου (Parameter-Shift Rule)**: $\partial C/\partial\theta_k = 0.5\,[\,C(\theta_k + \pi/2) - C(\theta_k - \pi/2)\,]$. Με επιστημονική ειλικρίνεια, τα σημερινά VQC αποτελούν ιδανικό ερευνητικό πεδίο για κβαντική χημεία (VQE) και βελτιστοποίηση (QAOA), ενώ η αντιμετώπιση των επίπεδων περιοχών μηδενικής κλίσης (*barren plateaus*) και η κλιμάκωση λογικών qubits αποτελούν τη γέφυρα προς την ανεκτική σε σφάλματα Κβαντική AI.



Σχήμα 17.3

Κβαντική απειλή, Μετακβαντική Κρυπτογραφία & Υβριδική Κβαντική AI: ο αλγόριθμος Shor έναντι των προτύπων πλεγμάτων/hash NIST (FIPS 203 ML-KEM, FIPS 204 ML-DSA, FIPS 205 SLH-DSA) και ο υβριδικός βρόχος VQC (QPU Ansatz U(θ) + κλασικός βελτιστοποιητής Parameter-Shift).

17.4 Πότε Συμπληρώνονται Κλασικές GPU και Κβαντικοί Επεξεργαστές (QPU)

Οι κβαντικοί υπολογιστές δεν πρόκειται να αντικαταστήσουν τις κλασικές CPU και GPU, αλλά να λειτουργήσουν ως εξειδικευμένοι επιστημονικοί επιταχυντές σε συνδυαστικές και κβαντομηχανικές υποκατηγορίες προβλημάτων:

Κριτήριο Σύγκρισης	Κλασικός Υπερυπολογιστής (GPU / TPU)	Κβαντικός Επεξεργαστής (QPU)
Μονάδα Πληροφορίας	Δυαδικό Bit (0 ή 1), δεκάδες χιλιάδες παράλληλοι πυρήνες	Qubit (υπέρθεση $\alpha\|0\rangle + \beta\|1\rangle$ & διεμπλοκή)
Ιδανικός Τομέας	Εκπαίδευση μεγάλων LLMs, CNN, επεξεργασία μεγάλων δεδομένων	Προσομοίωση μορίων, κβαντική χημεία, βελτιστοποίηση, παραγοντοποίηση
Περιβάλλον Λειτουργίας	Θερμοκρασία δωματίου σε κέντρα δεδομένων ή προσωπικούς υπολογιστές	Κρυογενική ψύξη (~15 mK), παγιδευμένα ιόντα ή φωτονικά κυκλώματα
Κύρια Πρόκληση	Ενεργειακή κατανάλωση και εύρος ζώνης μνήμης (HBM)	Αποσυνοχή και κλιμάκωση Λογικών Qubits διόρθωσης σφαλμάτων

Πρακτικός Οδικός Χάρτης Μετάβασης στη Μετακβαντική Ασφάλεια

1. Κρυπτογραφική Απογραφή

Καταγραφή όλων των συστημάτων, TLS πιστοποιητικών, VPN και βάσεων δεδομένων που βασίζονται σήμερα σε RSA ή ελλειπτικές καμπύλες (ECC).

2. Υβριδική Ανταλλαγή Κλειδιών

Σταδιακή ενεργοποίηση υβριδικών συνδυασμών (**X25519 + ML-KEM**) σε περιηγητές και διακομιστές για άμεση θωράκιση επικοινωνιών.

3. Υπογραφές & Κρυπτο-Ευελιξία

Σχεδίαση αρχιτεκτονικών *crypto-agility* με πρότυπα **ML-DSA (FIPS 204)** και **SLH-DSA (FIPS 205)** ώστε η αναβάθμιση αλγορίθμων να γίνεται απρόσκοπτα.

◆ **Κρίσιμη Προτεραιότητα Ασφάλειας Σήμερα: Harvest Now, Decrypt Later**

Η μετάβαση στη Μετακβαντική Κρυπτογραφία (**ML-KEM / ML-DSA / SLH-DSA**) ξεκινά ήδη σήμερα, διότι ευαίσθητα κρυπτογραφημένα δεδομένα που υποκλέπτονται στο παρόν θα μπορούσαν να αποκρυπτογραφηθούν μελλοντικά μόλις ωριμάσουν οι ανεκτικοί σε σφάλματα κβαντικοί υπολογιστές.

ΚΕΦΑΛΑΙΟ 18

Ο Δρόμος προς την Εξειδίκευση: Μαθηματικά, Κώδικας & Έρευνα

Ο ολοκληρωμένος οδικός χάρτης 4 βαθμίδων από φοιτητή σε ανεξάρτητο ερευνητή AI: μαθηματικοί πυλώνες, συστήματα GPU/LLMs, μελέτες αποκοπής και επιστημονική αναπαραγωγιμότητα.

Μαθησιακοί Στόχοι & Θεωρητικό Πλαίσιο Ενότητας

- **Έξι Μαθηματικοί Πυλώνες:** Σύνδεση Γραμμικής Άλγεβρας, Πιθανοτήτων, Λογισμού, Βελτιστοποίησης, Αριθμητικής Ανάλυσης και Θεωρίας Γράφων με πραγματικά νευρωνικά στρώματα.
- **Πρόοδος 4 Βαθμίδων (Φοιτητής → Ερευνητής):** Δομημένο πλάνο κατάκτησης από καθαρή NumPy έως PyTorch/CUDA, τοπικά LLMs (LoRA/RAG/MCP) και επιστημονικές δημοσιεύσεις.
- **Ερευνητική Μεθοδολογία & Αναπαραγωγιμότητα:** Σχεδίαση αυστηρών μελετών αποκοπής (*Ablation Studies*), σταθεροί σπόροι τυχαιότητας και ανοιχτά αποθετήρια κώδικα.
- **Υγιεινή Benchmarks:** Αποφυγή διαρροής δεδομένων (*data contamination*), ορθή στατιστική σύγκριση και ειλικρινής αναφορά αρνητικών αποτελεσμάτων.

18.1 Οι Έξι Μαθηματικοί Πυλώνες της Βαθιάς Μάθησης

ΠΡΩΤΑ ΒΗΜΑΤΑ & ΔΙΑΙΣΘΗΣΗ

Οδικός Χάρτης Πρώτων 90 Ημερών Αυτομελέτης (Από Μηδενική Εμπειρία έως Πλήρη Συστήματα AI)

Μην αφήσετε τον πίνακα των έξι μαθηματικών κλάδων να σας τρομάξει! Κανένας ερευνητής δεν κατέκτησε όλα τα μαθηματικά προτού γράψει την πρώτη του γραμμή κώδικα: η καλύτερη μέθοδος είναι η **παράλληλη μάθηση μέσα από εκτελέσιμα εργαστήρια**. Ακολουθήστε αυτό το ρεαλιστικό πρόγραμμα **90 ημερών (12 εβδομάδων)** με ρυθμό 5 έως 7 ώρες την εβδομάδα:

- **Εβδομάδες 1 έως 4 (Θεμέλια Python, NumPy & Εργαστήρια 01 έως 08):** Εξοικείωση με Python 3.12+, πίνακες NumPy, διαίσθηση παραγώγων και υλοποίηση των πρώτων νευρώνων και πυλών AND/XOR από το μηδέν.
- **Εβδομάδες 5 έως 8 (PyTorch/Keras, GPU & Εργαστήρια 09 έως 22):** Μετρικές αξιολόγησης, επιτάχυνση σε GPU/CUDA, Όραση (MNIST/CIFAR-10), Γλώσσα (BiLSTM/Transformers), πρόγνωση καιρού Πάτρας και Ενισχυτική Μάθηση (DQN).
- **Εβδομάδες 9 έως 12 (Τοπικά LLMs, LoRA, RAG, SETI & Εργαστήρια 23 έως 33):** Διαμέριση BPE, Mini-GPT, κβαντοποίηση 4-bit με Ollama, προσαρμογείς LoRA, διανυσματικές βάσεις RAG, πράκτορες MCP και ανάλυση ραδιοαστρονομικών σημάτων SETI!

Για να εργαστεί κανείς ουσιαστικά ως μηχανικός ή ερευνητής στον χώρο της Τεχνητής Νοημοσύνης και της Βαθιάς Μηχανικής Μάθησης, η βαθιά κατανόηση του μαθηματικού υπόβαθρου μετατρέπει τα εργαλεία από «μαύρα κουτιά» σε διαφανή επιστημονικά όργανα:

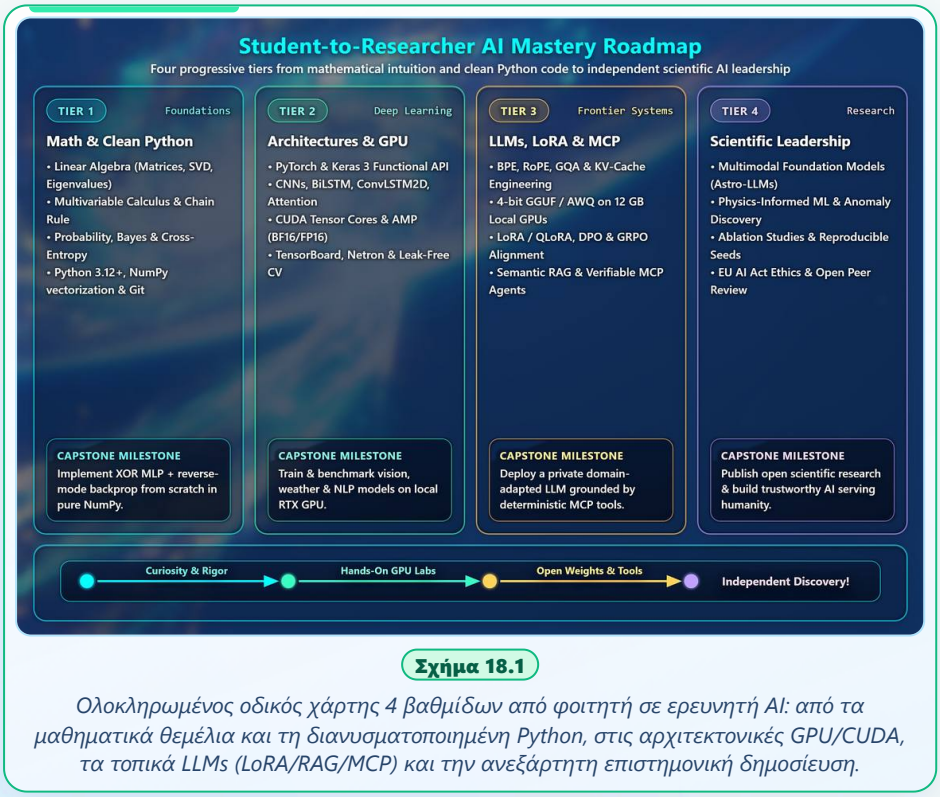
Μαθηματικός Κλάδος	Κεντρικές Έννοιες	Πού Εφαρμόζεται Άμεσα στην AI
1. Γραμμική Άλγεβρα	Διανύσματα, πίνακες, τανυστές, ιδιοτιμές, SVD	Πολλαπλασιασμοί επιπέδων, Embeddings, Self-Attention (Q, K, V), LoRA
2. Πιθανότητες & Στατιστική	Κατανομές, θεώρημα Bayes, εντροπία, KL divergence	Cross-Entropy, Softmax, Diffusion Models, DPO/RLHF, αβεβαιότητα
3. Απειροστικός Λογισμός	Μερικές παράγωγοι, κλίση (∇), Ιακωβιανή, κανόνας αλυσίδας	Οπισθοδιάδοση σφάλματος (Backpropagation), αυτόματη διαφόριση

Μαθηματικός Κλάδος	Κεντρικές Έννοιες	Πού Εφαρμόζεται Άμεσα στην AI
4. Βελτιστοποίηση & Αλγόριθμοι	Κυρτότητα, SGD, Adam/AdamW, δυναμικός προγραμματισμός	Σύγκλιση εκπαίδευσης, χρονοπρογραμματισμός lr, εξίσωση Bellman (RL)
5. Αριθμητικές Μέθοδοι	Αριθμητική ευστάθεια, FP16/BF16/INT4, Monte Carlo	Ασφαλές Softmax, μικτή ακρίβεια GPU, κβαντοποίηση NF4/GGUF
6. Θεωρία Γράφων	Κατευθυνόμενοι άκυκλοι γράφοι (DAG), HNSW, ροές	Υπολογιστικοί γράφοι (Functional API, Netron), GNNs, διανυσματικά ευρετήρια

18.2 Ο Οδικός Χάρτης 4 Βαθμίδων: Από Φοιτητή σε Ανεξάρτητο Ερευνητή AI

Η κατάκτηση της σύγχρονης AI επιτυγχάνεται καλύτερα μέσα από τέσσερις διαδοχικές, μετρήσιμες βαθμίδες εξειδίκευσης, όπου κάθε βαθμίδα ολοκληρώνεται με ένα απτό κατασκευαστικό ορόσημο:

1. **Βαθμίδα 1 (Μαθηματικά Θεμέλια & Καθαρή Python):** Γραμμική Άλγεβρα (πίνακες, SVD, ιδιοτιμές), Λογισμός Πολλών Μεταβλητών & Κανόνας Αλυσίδας, Πιθανότητες/Bayes/Εντροπία, διανυσματοποιημένη **NumPy** σε Python 3.12+ και έλεγχος εκδόσεων με **Git**. *Ορόσημο:* Υλοποίηση MLP XOR και οπισθοδιάδοσης από το μηδέν σε καθαρή NumPy (Εργαστήρια 02 & 06).
2. **Βαθμίδα 2 (Βαθιές Αρχιτεκτονικές, PyTorch/CUDA & Συστήματα):** PyTorch & Keras 3 Functional API, CNN, BiLSTM, ConvLSTM2D, Self-Attention, αξιοποίηση **CUDA Tensor Cores** & Μικτής Ακρίβειας AMP (BF16/FP16), ανάλυση με TensorBoard/Netron και αυστηρή διασταυρούμενη επικύρωση χωρίς διαρροή δεδομένων. *Ορόσημο:* Εκπαίδευση και μέτρηση απόδοσης μοντέλων όρασης, πρόγνωσης καιρού και NLP σε τοπική GPU (Εργαστήρια 07 έως 22).
3. **Βαθμίδα 3 (Μετωπικά LLMs, LoRA, RAG & Πράκτορες MCP):** Εμβάθυνση σε BPE, RoPE, GQA & μηχανική KV-Cache, κβαντοποίηση 4-bit (`GGUF/AWQ`) σε 12 GB GPU, εξειδίκευση **LoRA / QLoRA**, ευθυγράμμιση **DPO / GRPO**, υβριδικό RAG (Dense + BM25 + RRF) και ασφαλείς πράκτορες **MCP**. *Ορόσημο:* Ανάπτυξη ιδιωτικού, τεκμηριωμένου LLM συνδεδεμένου με ντετερμινιστικά εργαλεία (Εργαστήρια 23 έως 29).
4. **Βαθμίδα 4 (Επιστημονική Ηγεσία & Ανοιχτή Έρευνα):** Πολυτροπικά θεμελιώδη μοντέλα (όπως Astro-LLMs και Spectral ViT στα Εργαστήρια 30 έως 33), Φυσικά-Ενημερωμένη Μηχανική Μάθηση (PINNs), ανίχνευση άγνωστων ανωμαλιών, ηθική συμμόρφωση EU AI Act και δημοσίευση σε διεθνή συνέδρια και περιοδικά με ανοιχτή αξιολόγηση από ομοτίμους (*open peer review*).



Σχήμα 18.1

Ολοκληρωμένος οδικός χάρτης 4 βαθμίδων από φοιτητή σε ερευνητή AI: από τα μαθηματικά θεμέλια και τη διανυσματοποιημένη Python, στις αρχιτεκτονικές GPU/CUDA, τα τοπικά LLMs (LoRA/RAG/MCP) και την ανεξάρτητη επιστημονική δημοσίευση.

18.3 Επιστημονική Μεθοδολογία: Αναπαραγωγιμότητα, Μελέτες Αποκοπής & Υγιεινή Benchmarks

Η διαφορά ανάμεσα σε ένα απλό εκπαιδευτικό πείραμα και μία δημοσιεύσιμη επιστημονική συνεισφορά έγκειται στην αυστηρή πειραματική πειθαρχία. Κάθε διπλωματική, μεταπτυχιακή ή διδακτορική εργασία στην AI οφείλει να τηρεί τέσσερις αδιαπραγμάτευτους κανόνες ερευνητικής μηχανικής:

1. Ανοιχτή Αναπαραγωγιμότητα (Reproducibility)

Κλείδωμα σταθερών σπόρων τυχαιότητας (NumPy, PyTorch, CUDA), ακριβείς εκδόσεις πακέτων σε `requirements.txt`, καταγραφή υλικού (GPU/VRAM) και δημοσίευση καθαρού, αυτοτελούς κώδικα σε Git ώστε κάθε ερευνητής να αναπαράγει τους πίνακες με μία εντολή.

2. Μελέτες Αποκοπής (Ablation Studies)

Απομόνωση της αιτιώδους συνεισφοράς *κάθε* επιμέρους εξαρτήματος (π.χ. αφαίρεση RoPE, μεταβολή τάξης LoRA *r* ή απενεργοποίηση BM25 στο RAG) ένα στοιχείο κάθε φορά απέναντι σε ισχυρό μοντέλο αναφοράς (baseline), με αναφορά μέσου όρου ± τυπικής απόκλισης σε πολλαπλούς σπόρους.

3. Υγιεινή Προτύπων Αξιολόγησης (Benchmark Hygiene)

Αυστηρός έλεγχος για *μόλυνση δεδομένων δοκιμής (data contamination)*, χρήση χρονικών ή ομαδοποιημένων διαχωρισμών (όπως στα Εργαστήρια 16 έως 18 για τον καιρό), επαγρύπνηση απέναντι στον Νόμο του Goodhart (όταν μια μετρική γίνεται στόχος παύει να είναι αξιόπιστη μετρική) και στατιστικοί έλεγχοι σημαντικότητας.

4. Κριτική Σκέψη & Δια Βίου Μάθηση

Ειλικρινής τεκμηρίωση των αποτυχιών και των ορίων κάθε μεθόδου (όχι μόνο των επιτυχιών), συστηματική ανάγνωση πρωτογενών εργασιών (`arXiv`) και διαρκής πειραματική επαλήθευση κάθε θεωρητικού ισχυρισμού σε πραγματικό κώδικα.

ΑΡΙΘΜΗΤΙΚΟ ΠΑΡΑΔΕΙΓΜΑ ΒΗΜΑ ΠΡΟΣ ΒΗΜΑ

Πώς Συντάσσεται μια Καθαρή Μελέτη Αποκοπής (3-Seed Ablation Study) στην Επιστημονική Έρευνα

Για να αποδείξετε σε μια διπλωματική εργασία ή διεθνές συνέδριο ότι η μέθοδός σας όντως βελτιώνει την απόδοση (και δεν στάθηκε απλώς τυχερή σε μία εκτέλεση), προσθέτετε **ένα μόνο στοιχείο ανά γραμμή** και τρέχετε κάθε παραλλαγή σε **3 σταθερούς σπόρους τυχαιότητας** (`seeds = 42, 123, 2026`):

Παραλλαγή Μοντέλου (1 Αλλαγή ανά Βήμα)	3 Εκτελέσεις (Seeds 42, 123, 2026)	Μέσος Όρος ± Τυπ. Απόκλιση	Αιτιώδες Κέρδος (Δ)
1. Baseline CNN (FP32, χωρίς ABACAD)	87.8%, 88.2%, 88.6%	**88.2% ± 0.4%**	Βάση Αναφοράς
2. + AMP (BF16) & Spectral ViT	94.3%, 94.6%, 94.9%	**94.6% ± 0.3%**	**+6.4%**
3. + Φίλτρο ABACAD (Πλήρης Αγωγός)	97.2%, 97.4%, 97.6%	**97.4% ± 0.2%**	**+2.8%** (Σύνολο +9.2%)

Γιατί αυτός ο πίνακας πείθει κάθε κριτή: (α) Η διακύμανση μεταξύ σπόρων (*±0.2% έως ±0.4%*) είναι αμελητέα μπροστά στο κέρδος **+6.4%** (16× μεγαλύτερο από τον θόρυβο εκκίνησης!), (β) κάθε γραμμή αποδεικνύει χωριστά την αξία τόσο της αρχιτεκτονικής *ViT* όσο και του φυσικού φίλτρου *ABACAD*.

Συνοπτική Πυξίδα για τον Νέο Ερευνητή

Μην αντιμετωπίζετε τα μεγάλα μοντέλα ως μαγεία: ξεκινήστε υλοποιώντας μικρά δίκτυα από το μηδέν (όπως τα εργαστήρια `02`, `06`, `24` και `27` του `code.zip`), παρατηρήστε πώς κινούνται τα βάρη στο TensorBoard και το H5Web, και χτίστε βήμα βήμα τη δική σας ερευνητική διαίσθηση.

ΕΠΙΛΟΓΟΣ

«Η Άδεια Θέση» & Ανθρωπότητα και Νοημοσύνη προς το Σύμπαν

Πέρα από τους αλγορίθμους και την υπολογιστική ισχύ: η ανθρώπινη παρουσία, η ενσυναίσθηση και η ευθύνη μας απέναντι στο μέλλον.

Ε.1 Η Άδεια Θέση: Όταν η Γνώση Συναντά την Ανθρωπιά

Σε ολόκληρο το ταξίδι αυτού του σεμιναρίου εξερευνήσαμε συναπτικά βάρη, κλίσεις, συνελίξεις, μηχανισμούς αυτο-προσοχής, κάρτες GPU και κβαντικές πύλες. Είδαμε μηχανές να αναγνωρίζουν εικόνες, να μεταφράζουν γλώσσες, να συνθέτουν μουσική και να αγγίζουν ακόμη και τον συμβολισμό ενός ποιήματος. Όμως, στο τέλος κάθε επιστημονικής διαδρομής, αξίζει να σταθούμε σιωπηλοί μπροστά σε ένα ερώτημα που κανένας αλγόριθμος δεν μπορεί να απαντήσει αντί για εμάς.



Σχήμα Ε.1 *«Η Άδεια Θέση» | Τι αξίζει να γνωρίζουμε τα πάντα, αν δεν μάθουμε να ζούμε μαζί;*

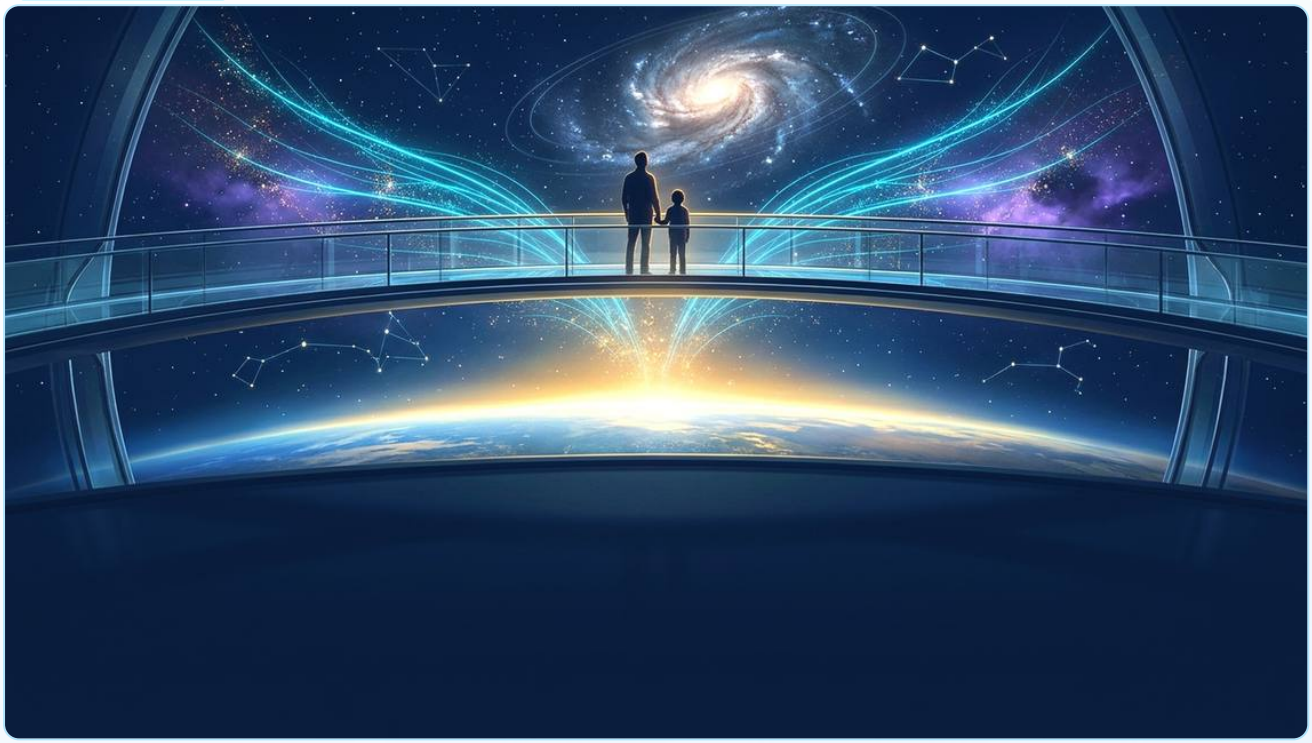
> *«Τι αξίζει να γνωρίζουμε τα πάντα, αν δεν μάθουμε να ζούμε μαζί;»*
>
> Η τεχνολογία πολλαπλασιάζει τη γνώση, όμως μόνο η ανθρώπινη συνύπαρξη, η ενσυναίσθηση και η φροντίδα της δίνουν νόημα.

Οι αλγόριθμοι μπορούν να επεξεργαστούν τρισεκατομμύρια παραμέτρους, αλλά δεν γνωρίζουν τη ζεστασιά μιας ανθρώπινης παρουσίας, το βάρος μιας υπόσχεσης ή τη χαρά της μοιρασμένης αναζήτησης. Καθήκον μας, ως πανεπιστημιακή κοινότητα, μηχανικοί και πολίτες, είναι να φροντίζουμε ώστε η Τεχνητή Νοημοσύνη να μην αδειάζει τις ανθρώπινες θέσεις γύρω μας, αλλά να υπηρετεί την αξιοπρέπεια, τη δικαιοσύνη και την ενότητα των ανθρώπων.

Ε.2 Ανθρωπότητα και Νοημοσύνη προς το Σύμπαν

Όταν η επιστημονική περιέργεια συμπορεύεται με την ηθική ευθύνη, η Τεχνητή Νοημοσύνη γίνεται γέφυρα ανάμεσα στις γενιές: ένα εργαλείο που βοηθά την ανθρωπότητα να θεραπεύσει ασθένειες, να προστατεύσει τον εύθραυστο πλανήτη μας και να στρέψει το βλέμμα της με θαυμασμό και ταπεινότητα προς τα άστρα.

Αυτό το σεμινάριο δεν αποτελεί το τέλος μιας διαδρομής, αλλά την αφετηρία για δικές σας ερωτήσεις, δικά σας πειράματα και νέες ιδέες. Μελετήστε τους αλγορίθμους με ακρίβεια, εκτελέστε και τροποποιήστε τους κώδικες του `code.zip`, αμφισβητήστε δημιουργικά όσα γνωρίζετε και κρατήστε πάντα στο κέντρο της σκέψης σας τον άνθρωπο.



Σχήμα Ε.2 *Ανθρωπότητα και Νοημοσύνη προς το Σύμπαν | Με πυξίδα τη γνώση, την ηθική και το φως της έρευνας.*

«Η γνώση φωτίζει τον δρόμο, αλλά η ανθρωπιά καθορίζει τον προορισμό.»

Σεμινάριο Τεχνητής Νοημοσύνης, Μηχανικής Μάθησης & Βαθιάς Μάθησης • Πανεπιστήμιο Πελοποννήσου

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών
ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

Ευχαριστίες & Κλείσιμο Εγχειριδίου

Αφιερωμένο στο προσωπικό και τους φοιτητές του Πανεπιστημίου Πελοποννήσου, καθώς και σε όλους όσοι αναζητούν την αλήθεια του κόσμου μέσα από την έρευνα, την περιέργεια και τη δημιουργική έκφραση.

Ευάγγελος Β. Πεφάνης | Μέλος Ε.Τ.Ε.Π., Τμήμα Η.Μ.Μ.Υ., Πανεπιστήμιο Πελοποννήσου

Συνοδευτικός Εργαστηριακός Κώδικας: `code.zip` (33 Εργαστήρια Python 3.12+)