



Analiza și Prognoza Seriilor de Timp

Capitolul 11: LLM-uri și Modele Fundaționale pentru Serii de Timp



Daniel Traian PELE

Academia de Studii Economice din București

IDA Institute Digital Assets

Blockchain Research Center

AI4EFin Artificial Intelligence for Energy Finance

Academia Română, Institutul de Prognoză Economică

MSCA Digital Finance

Obiective de învățare

La finalul acestui capitol, veți fi capabili să:

1. **Explicați** arhitectura Transformer și mecanismul de auto-atenție
2. **Distingeți** arhitecturile encoder-only, decoder-only și encoder-decoder
3. **Descrieți** paradigma modelelor fundamentale: pre-antrenare, ajustare fină, implementare
4. **Comparați** strategiile de tokenizare: patching, cuantizare și embedding-uri
5. **Evaluați** principalele modele fundamentale (Chronos, TimesFM, TimeGPT, Lag-Llama, Moirai)
6. **Decideți** când să utilizați inferența zero-shot vs. ajustarea fină
7. **Aplicați** modele fundamentale la sarcini reale de prognoză în Python
8. **Evaluați critic** limitările, preocupările etice și direcțiile viitoare

Cuprins

- Motivație
- Arhitectura Transformer
- De la Transformers la Modele Fundaționale
- Analiza Detaliată a Modelelor
- Zero-Shot vs Fine-Tuning
- Studii de Caz Practice
- Benchmark-uri și Selecția Modelelor
- Limitări, Etică și Viitor
- Utilizare IA
- Rezumat
- Quiz

De la modele specifice la modele fundamentale

Era 1: Modele statistice clasice (1970–2010)

- ▣ ARIMA, ETS, GARCH — construite manual per serie, ipoteze stricte
- ▣ Puncte forte: interpretabile, teorie bine înțeleasă, funcționează cu date puține

Era 2: Deep Learning (2015–2022)

- ▣ LSTM, TCN, DeepAR, N-BEATS — antrenate pe seturi de date individuale
- ▣ Puncte forte: captează dinamici neliniare; Slăbiciune: necesită date etichetate mari per sarcină

Era 3: Modele Fundamentale (2023–prezent)

- ▣ TimeGPT, Chronos, TimesFM, Lag-Llama, Moirai — pre-antrenate pe miliarde de puncte temporale
- ▣ **Perspectivă cheie:** antrenare *o singură dată* pe date masive și diverse, implementare *oriunde* (zero-shot)

Avantajul scalei

Modele Fundaționale: Comparație de scală

Model	Parametri	Date pre-antrenare	Arhitectură	Zero-Shot?
TimeGPT	N/A (proprietary)	100B+ puncte	Transformer enc-dec	Da (API)
Chronos	20M–710M	84B puncte	T5 enc-dec	Da
TimesFM	200M	100B puncte	Decoder-only	Da
Lag-Llama	352M	Seturi multiple	LLaMA decoder	Da
Moirai	14M–311M	27B puncte	BERT encoder	Da
PatchTST	1M–10M	Specific sarcinii	Transformer enc	Nu (supervizat)

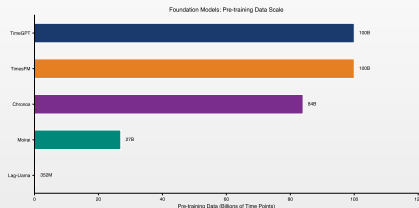
Observație cheie

Modelele fundamentale valorifică **transferul de cunoștințe între domenii**: tiparele învățate din date energetice ajută la prognoza vânzărilor cu amănuntul și invers. Aceasta este analogă modului în care GPT-3 învață gramatica din corpusuri textuale diverse.

Harta capitolului

Ce vom acoperi

1. **Motivație** — cele trei ere ale prognozei
2. **Arhitectura Transformer** — atenție, codificare pozițională
3. **De la Transformers la Modele Fundaționale** — tokenizare, patching
4. **Analiza Detaliată a Modelelor** — Chronos, TimesFM, TimeGPT, Lag-Llama, Moirai
5. **Zero-Shot vs Fine-Tuning** — când și cum
6. **Studii de Caz Practice** — EUR/RON, energie, volatilitate
7. **Benchmark-uri și Selecția Modelelor** — Monash, GiftEval
8. **Limitări, Etică și Viitor** — ce ne rezervă viitorul



Scala datelor de pre-antrenare pentru modelele fundamentale (scală logaritmică).

Cercetător în lumina reflectoarelor: Vaswani et al. (2017)

“Attention Is All You Need”

- ▣ Publicat la NeurIPS 2017 de cercetători de la Google Brain
- ▣ Peste **130,000 de citări** — una dintre cele mai influente lucrări ML din toate timpurile
- ▣ Proiectat inițial pentru **traducere automată** (NLP)
- ▣ Inovația cheie: a înlocuit complet recurența cu **auto-atenția**

De ce contează pentru seriile de timp

- ▣ Seriile de timp sunt date **secvențiale** — aceeași provocare fundamentală ca în limbaj
- ▣ Transformerul gestionează secvențe de **lungime variabilă** în mod natural
- ▣ **Calcul paralel**: spre deosebire de RNN-uri, toate pozițiile sunt procesate simultan
- ▣ Modelele fundamentale (GPT, BERT, T5) sunt toate construite pe această arhitectură

Cronologie de impact

2017: Attention Is All You Need → 2018: BERT, GPT → 2020: GPT-3 → 2023: TimeGPT, Chronos → 2024: TimesFM, Moirai

Auto-atenția: Intuiție

De ce atenția înlocuiește recurența

- ▣ **RNN-uri:** procesează token-urile secvențial $\Rightarrow O(T)$ pași secvențiali, blocaj informațional
- ▣ **Auto-atenția:** fiecare poziție acordă atenție fiecărei alte poziții **în paralel**
- ▣ Conexiuni directe între poziții îndepărtate: fără gradienti care se estompează

RNN: Secvențial

- ▣ Calea între x_1 și x_T : $O(T)$ pași
- ▣ Nu se poate paraleliza temporal
- ▣ Gradientul: T multiplicări
- ▣ Informația se “comprimă” prin starea ascunsă

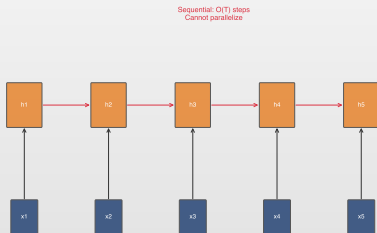
Atenție: Paralel

- ▣ Calea între oricare două poziții: $O(1)$
- ▣ Complet paralelizabilă pe GPU-uri
- ▣ Fiecare poziție “vede” direct toate celelalte
- ▣ Învăță la care poziții să acorde atenție

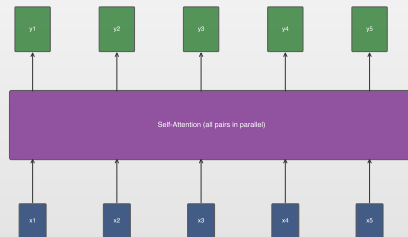
Atenția de sine: Intuiție (Diagramă)

RNN vs Transformer: Processing Paradigm

(a) RNN: Sequential Processing



(b) Transformer: Parallel Attention



RNN procesează token-urile secvențial ($O(T)$ pași); Transformer-ul atenționează toate pozițiile în paralel ($O(1)$ lungimea căii).

Auto-atenția: Matematica

Atenția Dot-Product Scalată

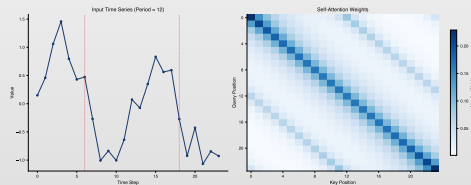
Dați fiind embedding-urile de intrare $X \in \mathbb{R}^{T \times d}$, calculăm:

$$Q = XW^Q, \quad K = XW^K, \quad V = XW^V$$

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

Interpretare

- Q (Queries): "Ce caut?"
- K (Keys): "Ce conțin?"
- V (Values): "Ce informație furnizez?"
- $\sqrt{d_k}$: scalarea previne saturarea softmax
- leșire: media ponderată a valorilor, ponderi = scoruri de compatibilitate



Harta termică a ponderilor de atenție: rândurile acordă atenție coloanelor.

Atenția Multi-Cap

Capete multiple de atenție captează tipare diverse

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \dots, \text{head}_h) W^O$$

where $\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$

Fiecare cap proiectează într-o dimensiune mai mică: $W_i^Q, W_i^K \in \mathbb{R}^{d \times d_k}$, $W_i^V \in \mathbb{R}^{d \times d_v}$ cu $d_k = d_v = d/h$.

Ce învață diferitele capete

- ▣ Capul 1: **Trend** — atenție la valorile recente
- ▣ Capul 2: **Sezonalitate** — atenție la aceeași perioadă din ciclul anterior
- ▣ Capul 3: **Valori aberante** — atenție la valorile extreme
- ▣ Capul 4: **Dependențe pe termen lung** — atenție la corelații îndepărtate

Cost computațional

- ▣ Complexitatea auto-atenției: $O(T^2 \cdot d)$
- ▣ Memorie: $O(T^2)$ pentru matricea de atenție
- ▣ Secvențe lungi ($T > 4096$): costul pătratic devine prohibitiv
- ▣ Soluții: patching, atenție rară, atenție liniară

Codificarea pozițională

De ce contează poziția

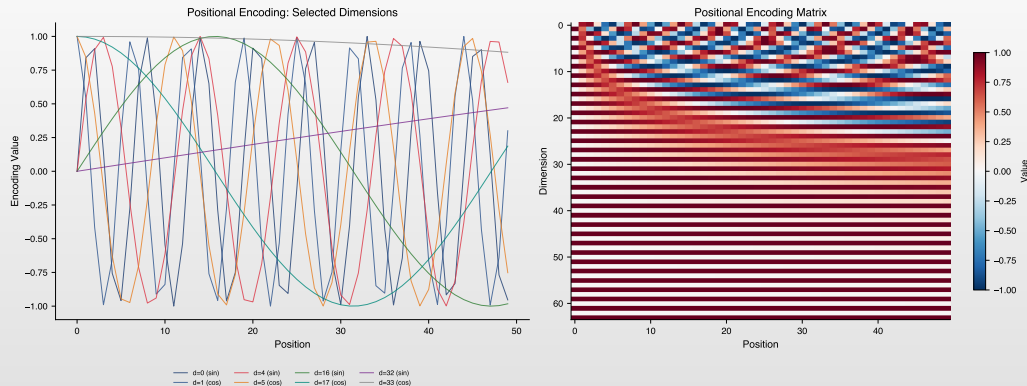
Auto-atenția este **echivariată la permutări**: fără informație pozițională, tratează $\{x_1, x_2, x_3\}$ și $\{x_3, x_1, x_2\}$ identic. Ordinea în seriile de timp **contează**, deci injectăm poziția.

Codificarea pozițională sinusoidală (Vaswani et al., 2017)

$$PE_{(pos, 2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right)$$
$$PE_{(pos, 2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right)$$

Fiecare dimensiune sinusoidală are o frecvență diferită $\Rightarrow$ semnătură pozițională unică. Pozițiile relative pot fi calculate dintr-un produs scalar.

Codificarea pozițională (Vizualizare)



Fiecare dimensiune oscilează la o frecvență diferită, creând o semnătură pozițională unică. Pozițiile relative pot fi calculate prin transformări liniare.

Blocul Transformer complet

Componentele blocului Transformer

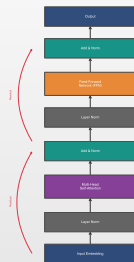
1. **Embedding de intrare** + Codificare pozițională
2. **Normalizare de strat** (variante pre-norm)
3. **Auto-atenție Multi-Head**
4. **Conexiune reziduală**: $x + \text{Attn}(x)$
5. **Normalizare de strat**
6. **Rețea Feed-Forward (FFN)**: două straturi liniare cu GELU/ReLU

$$\text{FFN}(x) = W_2 \cdot \text{GELU}(W_1 x + b_1) + b_2$$

7. **Conexiune reziduală**: $x + \text{FFN}(x)$

Stivuirea a N blocuri ($N = 6-24$) conferă adâncime. Conexiunile reziduale permit fluxul gradientului.

Transformer Block Architecture



Diagramă: blocul Transformer standard cu layout pre-norm.

Arhitecturi Encoder vs Decoder

Trei paradigme arhitecturale

	Encoder-only	Decoder-only	Encoder-Decoder
Origine NLP	BERT	GPT	T5, BART
Atenție	Bidirecțională	Cauzală (mascată)	Atenție încrucișată
Modele TS	Moirai, PatchTST	TimesFM, Lag-Llama	Chronos, TimeGPT
Puncte forte	Învăță reprezentări bogate	Natural pentru generare autoregresivă	Mapare flexibilă intrare-ieșire
Utilizare TS	Reprezentare, clasificare	Proгноză, generare	Proгноză, seq2seq

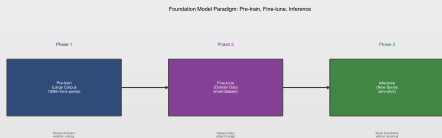
Distincție cheie

Mascarea cauzală (decoder): poziția t poate acorda atenție doar la $\{1, \dots, t\}$. Previne “privitul” în viitor — potrivire naturală pentru prognoza seriilor de timp. **Bidirecțional** (encoder): fiecare poziție vede toate celelalte — util pentru învățarea reprezentărilor și clasificare.

Paradigma modelelor fundamentale

Pipeline în trei etape

1. **Pre-antrenare:** Antrenare pe corpusuri masive și diverse (miliarde de puncte temporale din diferite domenii)
2. **Ajustare fină** (opțional): Adaptare la un domeniu specific cu o rată de învățare mică
3. **Inferență:** Implementare pe serii noi, nevăzute — zero-shot sau few-shot



Perspectivă cheie (Bommasani et al., 2021)

- “Un model fundamental este orice model antrenat pe date vaste care poate fi adaptat la sarcini downstream.”
- Pentru TS: antrenare pe *energie + finanțe + retail + sănătate + meteo* $\Rightarrow$ prognoză pentru *orice*
- Ipoteza critică: **tipare temporale comune** există între domenii

Strategii de tokenizare pentru serii de timp

Provocare: Valori continue $\rightarrow$ Token-uri discrete

Modelele NLP procesează token-uri discrete. Seriile de timp sunt continue $\Rightarrow$ necesită o strategie de mapare.

Strategia 1: Embedding-uri de valori brute

- ▣ Proiectează $x_t \in \mathbb{R}$ în $\mathbb{R}^d$ prin strat liniar
- ▣ Folosită de: modele TS Transformer vanilla

Strategia 2: Scalare + Cuantizare

- ▣ Normalizare, apoi mapare în B bin-uri discrete
- ▣ Folosită de: **Chronos** (4096 bin-uri)

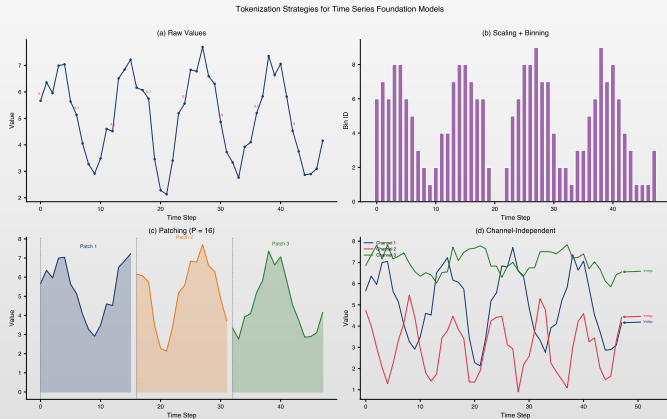
Strategia 3: Patching

- ▣ Grupează P puncte consecutive într-un token
- ▣ Folosită de: **PatchTST**, **TimesFM**

Strategia 4: Canale independente

- ▣ Fiecare variabilă procesată independent
- ▣ Folosită de: **PatchTST**, **Moirai**

Strategii de tokenizare (Vizualizare)



Patru abordări pentru conversia seriilor de timp continue în token-uri discrete pentru modelele Transformer.

Patching: De ce funcționează

Perspectiva PatchTST (Nie et al., 2023)

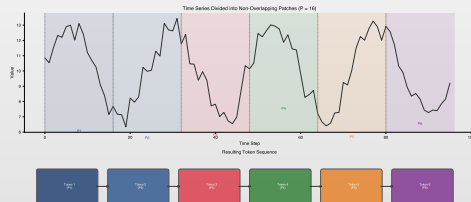
Se împarte o serie de timp de lungime L în $N = \lfloor L/P \rfloor$ patch-uri non-suprapuse de dimensiune P :

$$\mathbf{p}_i = (x_{(i-1)P+1}, x_{(i-1)P+2}, \dots, x_{iP}) \in \mathbb{R}^P, \quad i = 1, \dots, N$$

Fiecare patch este proiectat liniar la dimensiunea d și furnizat ca un singur token pentru Transformer.

Beneficii ale patching-ului

- ▣ **Reduce complexitatea:** atenție $O(T^2) \rightarrow O((T/P)^2)$
- ▣ **Semantică locală:** fiecare patch captează un sub-tipar scurt (ex. ciclul zilnic)
- ▣ **Context mai lung:** cu $P = 32$, 512 token-uri acoperă 16,384 pași temporali
- ▣ **Reprezentări mai bune:** token-urile patch sunt mai informative decât cele punct



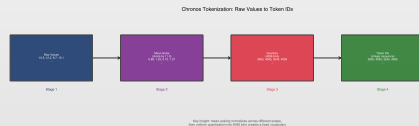
Cuantizarea: Abordarea Chronos

Tokenizare prin scalare și cuantizare (Ansari et al., 2024)

1. **Normalizare:** $\tilde{x}_t = \frac{x_t - \bar{x}}{s}$ unde $\bar{x}$ este media și s este media valorii absolute
2. **Cuantizare:** mapare în B bin-uri prin CDF: $b_t = \lfloor (B - 1) \cdot \Phi(\tilde{x}_t) \rfloor$ unde Φ este CDF-ul normal standard
3. **Embedding:** căutare bin b_t în tabelul de embedding T5 ($B = 4096$)

De ce cuantizăm?

- Reutilizează **mecanismele de vocabular NLP**: embedding-uri, strat de ieșire softmax
- Funcția de pierdere cross-entropy (categorială) este bine înțeleasă
- Probabilistică: ieșirea este o distribuție peste bin-uri $\Rightarrow$ distribuție predictivă completă
- Compromis: rezoluția limitată de numărul de bin-uri



Obiective de pre-antrenare

Paradigma 1: Predicția următorului token (Autoregresiv, stil GPT)

- ▣ Prezice x_{t+1} dat fiind $x_1, \dots, x_t$. Pierdere: $\mathcal{L} = -\sum_{t=1}^{T-1} \log p(x_{t+1} | x_1, \dots, x_t)$
- ▣ Folosită de: **TimesFM**, **Lag-Llama**, **TimeGPT**

Paradigma 2: Predicție mascată (stil BERT)

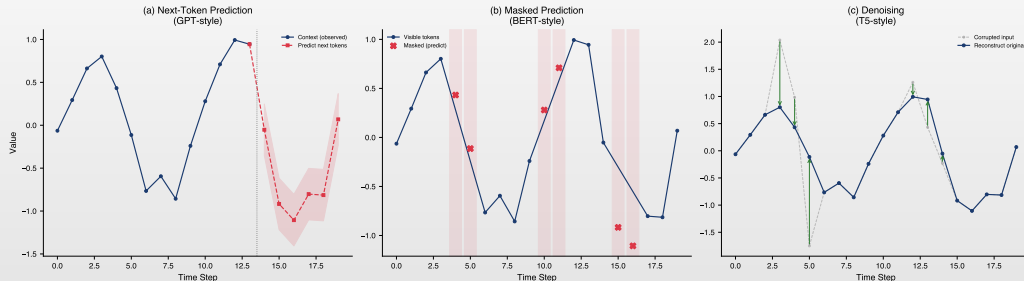
- ▣ Se maschează 15–40% din token-uri, se prezic din context. Pierdere:
 $\mathcal{L} = -\sum_{t \in \mathcal{M}} \log p(x_t | x_{\setminus \mathcal{M}})$
- ▣ Folosită de: **Moirai**, **PatchTST** (auto-supervizat)

Paradigma 3: Denoising / Seq2Seq (stil T5)

- ▣ Encoder procesează contextul, decoder generează prognoza. Pierdere: cross-entropy pe token-uri cuantizate
- ▣ Folosită de: **Chronos**

Obiective de pre-antrenare (Vizualizare)

Pre-training Objectives for Time Series Foundation Models



Trei paradigme de pre-antrenare: autoregresivă (stil GPT), predicție mascată (stil BERT) și denoising/seq2seq (stil T5).

Augmentarea datelor pentru pre-antrenare

KernelSynth (Ansari et al., 2024)

- Generează serii de timp **sintetice** prin compunerea de kernel-uri de Procese Gaussiene (GP):

$$k(t, t') = k_{\text{RBF}}(t, t') + k_{\text{periodic}}(t, t') + k_{\text{linear}}(t, t') + \dots$$

- Produce tipare realiste: trenduri, sezonabilitate, puncte de schimbare, zgomot
- Chronos folosește **miliarde** de serii sintetice pentru a augmenta datele reale

TSMix (TimesFM)

- Mixarea seriilor existente:
 $\tilde{x} = \lambda x_a + (1 - \lambda)x_b$
- Augmentează diversitatea fără a necesita expertiza GP
- Combinat cu tăierea ferestrelor și mixarea frecvențelor

De ce contează augmentarea

- Previne supraajustarea la domenii specifice
- Crește diversitatea tiparelor în timpul pre-antrenării
- Chronos: 50% date reale + 50% sintetice $\Rightarrow$ **mai bun** decât 100% reale

Chronos: Arhitectură

Chronos (Ansari et al., 2024 — Amazon)

- ▣ **Arhitectură:** T5 encoder-decoder (model de limbaj pre-antrenat)
- ▣ **Dimensiuni:** Mini (20M), Small (46M), Base (200M), Large (710M)
- ▣ **Tokenizare:** Normalizare medie-scală + cuantizare în **4096 bin-uri**
- ▣ **Pre-antrenare:** 84B puncte din 28 seturi publice + KernelSynth
- ▣ **Ieșire:** Distribuție categorială peste bin-uri $\Rightarrow$ prognoză probabilistică

Inovații cheie

- ▣ Prognoza ca **modelare de limbaj** peste token-uri cuantizate
- ▣ Nicio modificare arhitecturală a T5 — doar tokenizarea este nouă
- ▣ Probabilistic: eșantionează N traiectorii, calculează cuantilele
- ▣ Open-source, dimensiuni multiple pentru bugete diferite

Limitări

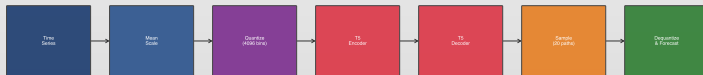
- ▣ Rezoluție limitată de 4096 bin-uri (eroare cuantizare)
- ▣ Doar univariat (fiecare variabilă separat)
- ▣ Fereastră de context: 512 token-uri
- ▣ Inferență mai lentă decât decoder-only (enc+dec)

Chronos: Cum prognozează

Pipeline pas cu pas

1. **Intrare:** seria de timp brută $(x_1, x_2, \dots, x_T)$
2. **Normalizare:** medie-scală $\tilde{x}_t = x_t / |\bar{x}|$
3. **Cuantizare:** mapează fiecare $\tilde{x}_t$ la unul din 4096 bin-uri prin Φ (CDF normal)
4. **Codificare:** encoder-ul T5 procesează token-urile de context $\rightarrow$ reprezentări ascunse
5. **Decodificare:** decoder-ul T5 generează token-uri de prognoză **autoregresiv**
6. **Eșantionare:** repetă decodificarea de N ori (ex. $N = 20$) cu eşantionare cu temperatură
7. **Decuantizare:** convertește indicii bin-urilor înapoi la valori continue
8. **Ieșire:** prognoza mediană + intervale cuantile (10%, 90%)

Chronos Inference Pipeline

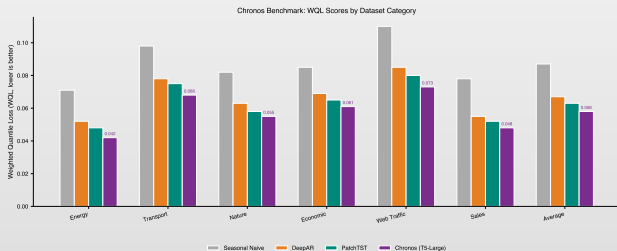


Probabilistic forecasting via categorical distribution over quantized bins

Chronos: Rezultate benchmark

Performanță pe benchmark-ul Monash (Ansari et al., 2024)

- ▣ Evaluat pe 27 seturi de date din Monash Time Series Forecasting Archive
- ▣ Metrici: **WQL** (Weighted Quantile Loss) și **MASE** (Mean Absolute Scaled Error)
- ▣ Chronos-Large obține cel mai bun sau al doilea rezultat pe 16/27 seturi de date **în zero-shot**
- ▣ Depășește modelele de deep learning specifice sarcinii (DeepAR, TFT, N-BEATS) în medie



Scoruri WQL: mai mic este mai bun. Chronos-Large competitiv cu sau mai bun decât bazele de referință supervizate.

TimesFM: Arhitectură

TimesFM (Das et al., 2024 — Google Research)

- ▣ **Arhitectură:** Transformer Decoder-only (200M parametri)
- ▣ **Patching:** dimensiune patch $P = 32$, proiectare la dimensiunea modelului
- ▣ **Pre-antrenare:** 100B puncte reale + sintetice
- ▣ **Multi-rezoluție:** frecvențe diverse (minutar la anual)
- ▣ **Context:** până la 512 patch-uri (= 16,384 pași temporali)

Inovații cheie

- ▣ **Decoder-only:** mai simplu și mai rapid decât enc-dec
- ▣ **Patch-uri de ieșire:** prezice P valori simultan
- ▣ **Token de frecvență:** codifică frecvența datelor
- ▣ Prognoză punctuală + ajustare fină opțională pt. cuantile

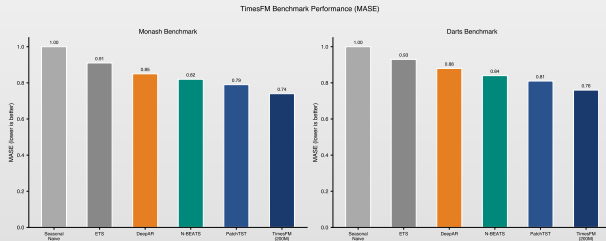
Comparație cu Chronos

- ▣ Valori continue (fără cuantizare) $\Rightarrow$ fără eroare de discretizare
- ▣ Inferență mai rapidă (decoder-only, ieșire pe patch-uri)
- ▣ Mai puțin probabilistic (necesită ajustare fină pt. cuantile)
- ▣ Ponderi închise până în 2024, acum parțial deschise

TimesFM: Performanță Zero-Shot

Rezultate benchmark (Das et al., 2024)

- Evaluat pe Monash, M3, M4 și benchmark-uri de orizont lung (ETTh, Weather, Electricity)
- **Zero-shot** TimesFM depășește modelele supervizate pe majoritatea seturilor Monash
- Deosebit de puternic pe prognoze de **orizont lung** (96–720 pași)
- Competitiv cu PatchTST și iTransformer specifice sarcinii, chiar fără ajustare fină



MAE zero-shot TimesFM pe seturile de date benchmark. Barele arată îmbunătățirea relativă față de cea mai bună bază de referință supervizată.

TimeGPT: Model fundațional API-First

TimeGPT (Garza & Mergenthaler-Canseco, 2024 — Nixtla)

- ▣ **Arhitectură:** Transformer encoder-decoder proprietar
- ▣ **Pre-antrenare:** cel mai mare corpus — **100B+** puncte temporale din domenii diverse
- ▣ **Acces:** doar API (fără ponderi deschise), produs comercial
- ▣ **Caracteristici:** zero-shot, intervale conformale, detecția anomaliilor, ajustare fină, variabile exogene

Avantaje

- ▣ **Cel mai ușor de folosit:** 3 linii de cod Python
- ▣ **Predicție conformală:** incertitudine calibrată
- ▣ **Variabile exogene:** spre deosebire de majoritatea FM
- ▣ Actualizări regulate de echipa Nixtla

Limitări

- ▣ **Cutie neagră:** proprietar, fără reproductibilitate
- ▣ **Cost API:** tarificare per apel
- ▣ **Confidențialitatea datelor:** trimise la serverele Nixtla
- ▣ **Dependență de furnizor:** disponibilitate serviciu

Lag-Llama: Bază LLM pentru serii de timp

Lag-Llama (Rasul et al., 2024)

- ▣ **Arhitectură:** decoder LLaMA adaptat pentru serii de timp (352M parametri)
- ▣ **Inovație cheie:** **Caracteristici lag ca embedding-uri de token-uri** — lag-uri $\{1, 2, \dots, 7, 14, 28, \dots\}$
- ▣ **Cap de distribuție:** produce parametri Student- t (μ, σ, ν) în loc de probabilități de token
- ▣ **Pre-antrenare:** Seturi de date de serii de timp disponibile public

De ce LLaMA pentru serii de timp?

- ▣ Baza LLaMA: dovedită la scară în NLP
- ▣ RoPE pentru poziții temporale
- ▣ GQA pentru inferență eficientă
- ▣ Token-urile lag captează structuri temporale (săptămânal, lunar)

Capul de distribuție

- ▣ În loc de softmax peste vocabular:
$$p(x_{t+1} \mid \cdot) = \text{Student-}t(\mu, \sigma, \nu)$$
- ▣ Gestionează natural cozile groase (date financiare)
- ▣ Probabilistic: eşantionare pt. intervale de predicție
- ▣ Open-source și ajustabil fin

Moirai: Orice-variabilă, orice-frecvență

Moirai (Woo et al., 2024 — Salesforce)

- ▣ **Arhitectură:** encoder mascat stil BERT (14M, 91M, 311M parametri)
- ▣ **Inovație:** gestionează **orice nr. de variabile, orice frecvență, orice lungime de predicție**
- ▣ **Pre-antrenare:** 27B observații din **LOTSA** (Large-scale Open Time Series Archive, 9 domenii)
- ▣ **Ieșire:** distribuție mixtură (parametrii unei distribuții de ieșire flexibile)

Design universal

- ▣ **Orice-variabilă:** aplatizează multivariatul în patch-uri, ID-uri de variabilă
- ▣ **Orice-frecvență:** embedding-uri de frecvență (orar, zilnic, etc.)
- ▣ **Orice-lungime:** atenție mascată pt. lungimi de ieșire variabile
- ▣ Cel mai apropiat de un FM TS “universal”

Setul de date LOTSA

- ▣ 9 domenii: energie, transport, finanțe, sănătate, natură, web, vânzări, climă, social
- ▣ 27B observații, curate pentru calitate
- ▣ Colecție de date open-source
- ▣ Permite pre-antrenare reproductibilă

PatchTST: Planul arhitectural

PatchTST (Nie et al., 2023)

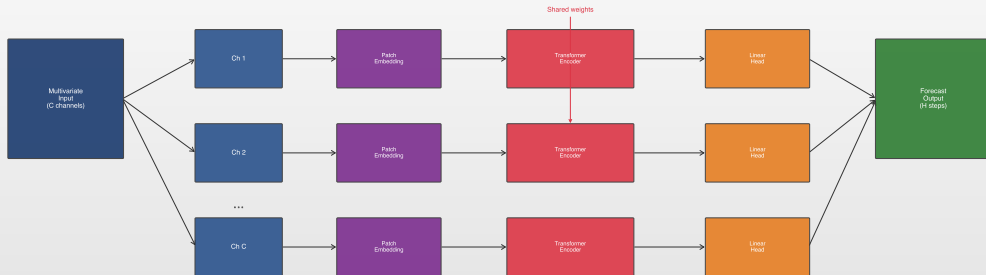
- ▣ **Arhitectură:** encoder Transformer cu **patching independent pe canale**
- ▣ A introdus două idei cheie adoptate de toate modelele ulterioare: **(1)** Patching: grupează P puncte temporale într-un singur token; **(2)** Independență pe canale: fiecare variabilă procesată independent
- ▣ Parametri: 1M–10M (specifici sarcinii, mult mai mici decât FM)

Moștenire

- ▣ PatchTST a arătat că Transformatorele pot depăși modelele liniare pe orizont lung
- ▣ Patching adoptat de: TimesFM, Moirai, MOMENT
- ▣ Independența pe canale: Moirai, iTransformer
- ▣ Pre-antrenare auto-supervizată: patching mascat

PatchTST: Diagramă de arhitectură

PatchTST Architecture: Channel-Independent Patched Transformer



PatchTST procesează fiecare canal independent, grupează pași de timp consecutivi în patch-uri și le transmite printr-un encoder Transformer partajat.

Tabelul comparativ al arhitecturilor

Comparație comprehensivă a modelelor

Model	Arhitectură	Parametri	Tokenizare	Ieșire	Pre-antrenare
Chronos	T5 enc-dec	20M–710M	Cuantizare (4096)	Dist. categorială	84B pct + KernelSynth
TimesFM	Decoder-only	200M	Patching ($P=32$)	Punct + cuantile	100B real + sintetic
TimeGPT	Enc-dec (prop.)	N/A	Proprietar	Punct + conformal	100B+ (proprietar)
Lag-Llama	LLaMA decoder	352M	Caracteristici lag	Student- t	Seturi publice
Moirai	BERT encoder	14M–311M	Patching + ID variabilă	Dist. mixtură	27B (LOTSAs)
PatchTST	Transformer enc	1M–10M	Patching ($P=16$)	Punct (MSE)	Specific sarcinii

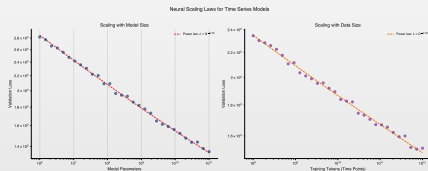
Concluzie cheie

Nicio arhitectură unică nu domină. Alegerea depinde de: (1) dacă aveți nevoie de prognoze probabilistice, (2) suport multivariat, (3) buget computațional și (4) dacă open-source contează.

Ipoteza scalării pentru serii de timp

Funcționează scalarea pentru serii de timp?

În NLP, legile de scalare (Kaplan et al., 2020): pierderea $\propto N^{-\alpha}$ unde N = parametri. Pentru serii de timp: dovezile sunt mixte.



De ce scalarea TS diferă de NLP

- **Mai puțină diversitate:** textul de pe internet > seturile TS
- **Tipare mai simple:** multe TS descrise de AR + sezonaliitate
- **Randamente descrescătoare:** Chronos-Large $\approx$ Chronos-Base
- **Calitatea datelor:** TS zgomotoase limitează ce pot învăța modelele
- Chronos 710M $\approx$ 200M; TimesFM 200M competitiv cu modele mari
- Calitatea datelor de pre-antrenare > cantitatea de parametri

Când modelele fundamentale eșuează

Moduri de eșec de monitorizat

1. **Schimbarea distribuției:** Schimbări de regim financiar (criza COVID, creșteri ale ratei dobânzii). Datele de pre-antrenare pot să nu acopere astfel de regimuri.
2. **Variabile exogene lipsă:** Vremea afectează cererea de energie; schimbările de politici afectează economia. Majoritatea FM sunt **univariate** și ignoră covariatele.
3. **Frecvențe rare:** Eșantionare neregulată, date conduse de evenimente, date tick-by-tick. Pre-antrenarea favorizează frecvențele regulate.
4. **Serii foarte scurte:** Cu < 50 observații, contextul este insuficient. Metodele clasice (ETS, ARIMA) domină adesea.
5. **Dependențe multivariate:** Corelațiile între variabile (ex. dinamici VAR) sunt ignorate de FM univariate. Doar Moirai încearcă suport multivariat.

Regulă practică

Dacă datele dvs. au structură causală cunoscută, caracteristici specifice domeniului sau schimbări de regim, combinați prognozele FM cu modele informate de domeniu (ansamblu sau ierarhic).

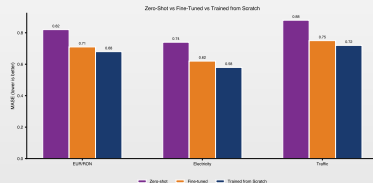
Zero-Shot: Când și de ce

Prognoză Zero-Shot

Se folosește modelul pre-antrenat direct pe o serie nouă fără niciun antrenament.

Când să folosiți Zero-Shot

- **Date limitate:** produs nou, pornire la rece
- **Prototipare rapidă:** bază de referință în minute
- **Cross-domeniu:** finanțe, retail, energie
- **Multe serii:** 10K+, fără antrenare individuală
- **Fără expertiză ML:** simplu apel API



Descoperire surprinzătoare

Ansari et al. (2024): Chronos zero-shot depășește modelele specifice (DeepAR, N-BEATS) *antrenate pe datele țintă*. Pre-antrenarea transferă cunoștințe temporale eficient.

Ajustare fină: Adaptare la domeniu

Ajustare fină completă vs LoRA

- ▣ **Ajustare fină completă:** actualizarea tuturor parametrilor pe date de domeniu (costisitoare, risc de uitare)
- ▣ **LoRA** (Low-Rank Adaptation, Hu et al., 2022): înghețarea ponderilor de bază, adăugarea de matrice mici antrenabile: $W' = W + \Delta W = W + BA$, $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times d}$, $r \ll d$. LoRA: 0.1–1% din parametri antrenabili, performanță similară, mult mai ieftină

Riscurile ajustării fine

- ▣ **Uitare catastrofală:** modelul pierde cunoștințele pre-antrenate
- ▣ **Supraajustare:** set de date mic $\Rightarrow$ generalizare slabă
- ▣ **Sensibilitate la hiperparametri:** rata de învățare, epoci critice

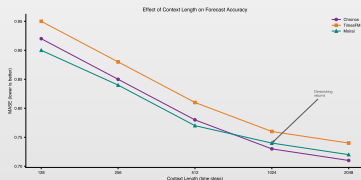
Bune practici

- ▣ **Rată de învățare mică** (10^{-5} – 10^{-4})
- ▣ **Încălzire** (500–1000 pași) + **oprire timpurie**
- ▣ **Înghețați straturile timpurii**, ajustați fin ulterioarele
- ▣ Preferați **LoRA** pentru seturi mici ($<10K$ serii)

Few-Shot și învățare în context

Lungimea contextului ca “Few-Shot” pentru serii de timp

- În NLP, “few-shot” = exemple în prompt. În TS, “context” = observații istorice
- **Context mai lung** → **proгноze mai bune** (până la fereastra de context a modelului)
- Modelul condiționează pe tiparele istorice pentru a deduce trend, sezonabilitate și zgomot



Eroare prognoză vs. lungimea contextului: randamente descrescătoare dincolo de fereastra modelului.

Ferestre de context per model

- Chronos: 512 token-uri (≈ 512 pași)
- TimesFM: 512 patch-uri ($\approx 16K$ pași)
- Moirai: 5000 token-uri (configurabil)
- Lag-Llama: 1024 pași (cu lag-uri extinse)
- TimeGPT: configurabil prin API

Sfat practic

Furnizați cât mai mult context permite modelul. Zilnic: ≥ 1 an.
Lunar: ≥ 5 ani.

Cadru decizional

Când să alegeți Zero-Shot, Ajustare Fină sau Clasic

- ▣ **Pasul 1:** Aveți date etichetate de domeniu? Nu → Zero-shot
- ▣ **Pasul 2:** Performanța zero-shot suficientă? Da → Implementare; Nu → Ajustare fină
- ▣ **Pasul 3:** Domeniul foarte diferit de pre-antrenare? Da → Ajustare fină completă; Nu → LoRA
- ▣ **Pasul 4:** Nevoie de interpretabilitate/cauzalitate? Da → Modele clasice (ARIMA, VAR)



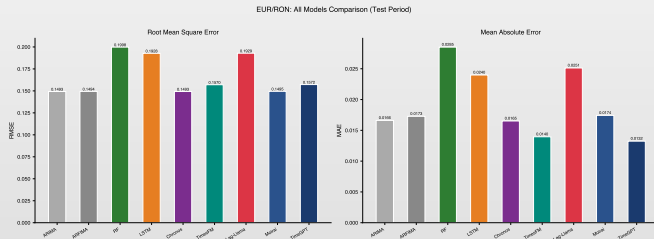
Reguli practice

- ▣ <50 observații: ARIMA / ETS
- ▣ 50–500 obs.: FM zero-shot
- ▣ 500–10K obs.: FM ajustat fin
- ▣ >10K obs.: DL specific poate egala FM
- ▣ Întotdeauna comparați cu ARIMA

Studiu de caz 1: EUR/RON cu toate modelele fundamentale

Extinderea studiului de caz din Capitolul 8

- ▣ **Date:** Cursul de schimb zilnic EUR/RON (2020–2024, ~1,200 observații)
- ▣ **Sarcină:** Prognoză la 30 de zile, separare temporală 80/20
- ▣ **Modele:** ARIMA, ARFIMA, RF, LSTM, **Chronos**, **TimesFM**, **Lag-Llama**, **Moirai**, **TimeGPT**
- ▣ Modelele fundamentale folosite în mod **zero-shot** (fără antrenare pe datele EUR/RON)



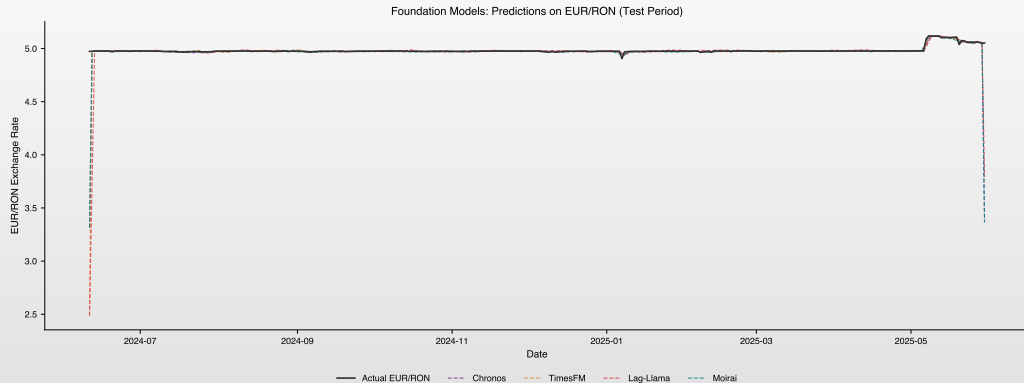
EUR/RON: Rezultate detaliate

Comparație performanță prognoză

Model	RMSE	MAE	MAPE (%)	Time antrenare
ARIMA(1,1,1)	—	—	—	< 1s
ARFIMA	—	—	—	< 1s
Random Forest	—	—	—	~2s
LSTM (2-layer)	—	—	—	~60s
Chronos-Base	—	—	—	0s (zero-shot)
TimesFM	—	—	—	0s (zero-shot)
Lag-Llama	—	—	—	0s (zero-shot)
Moirai-Base	—	—	—	0s (zero-shot)
TimeGPT	—	—	—	0s (apel API)

Notă: Valorile metricilor (—) sunt populate la rulare din codul Quantlet. Consultați notebook-ul companion pentru numere exacte.

EUR/RON: Comparație predicții

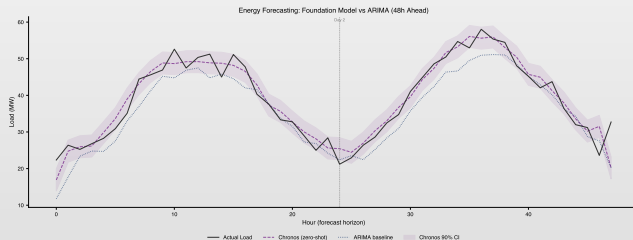


Predicțiile modelelor fundamentale suprapuse pe cursul EUR/RON. Toate modelele fundamentale folosesc inferență zero-shot.

Studiu de caz 2: Prognoza cererii de energie

Cererea de electricitate din Germania (orar, multi-sezonier)

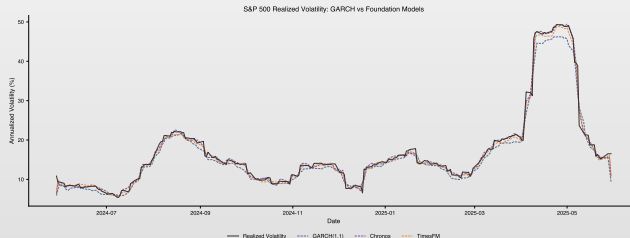
- ▣ **Date:** Consumul orar de electricitate din Germania (2015–2023, 70K+ observații)
- ▣ **Provocare:** sezonaliități multiple (24h zilnic + 168h săptămânal + anual)
- ▣ **Modele:** TBATS (Cap9), Prophet (Cap9), LSTM (Cap8), Chronos, TimesFM, Moirai
- ▣ Modelele fundamentale gestionează tiparele multi-sezoniere **fără specificare explicită**



Studiu de caz 3: Volatilitate financiară

Volatilitatea realizată S&P 500

- ▣ **Date:** Volatilitatea realizată zilnică S&P 500 (2000–2024)
- ▣ **Provocare:** Grupare, cozi groase, schimbări de regim (criza din 2008, COVID 2020)
- ▣ **Modele:** GARCH(1,1) (Cap5), HAR-RV, Chronos, TimesFM, Lag-Llama
- ▣ **Întrebare:** Pot modelele fundamentale capta dinamica de volatilitate fără structura GARCH?



Tipare de cod Python

Chronos (3 linii)

```
from chronos import ChronosPipeline
pipe = ChronosPipeline.from_pretrained(
    "amazon/chronos-t5-base")
forecast = pipe.predict(
    context=torch.tensor(data),
    prediction_length=30)
```

TimesFM (3 linii)

```
import timesfm
tfm = timesfm.TimesFm(hparams=
    timesfm.TimesFmHparams())
tfm.load_from_checkpoint()
preds = tfm.forecast(data)
```

TimeGPT (3 linii)

```
from nixtla import NixtlaClient
nixtla = NixtlaClient(api_key="...")
fcst = nixtla.forecast(
    df=df, h=30, model="timegpt-1")
```

Lag-Llama (4 linii)

```
from lag_llama.gluon.estimator import
    LagLlamaEstimator
estimator = LagLlamaEstimator(
    prediction_length=30)
predictor = estimator.train(train_ds)
forecasts = list(
    predictor.predict(test_ds))
```

Benchmark-uri majore

Monash Time Series Forecasting Archive

- ▣ **30K+** serii din 30 seturi de date (turism, electricitate, trafic, meteo, finanțe, etc.)
- ▣ Protocol standard de evaluare: separări fixe antrenare/test, metrici WQL și MASE
- ▣ Cel mai utilizat benchmark pentru modelele fundamentale (Chronos, TimesFM, Moirai raportează aici)

Competițiile M

- ▣ **M3** (2000): 3,003 serii, lunar/trimestrial/anual
- ▣ **M4** (2018): 100,000 serii, 6 frecvențe
- ▣ **M5** (2020): Vânzări Walmart, ierarhice
- ▣ Constatare: metodele mai simple deseori competitive

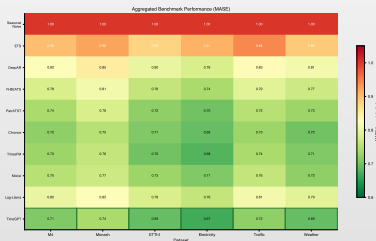
GiftEval (2024)

- ▣ **General Time Series Forecasting Evaluation**
- ▣ Proiectat special pentru evaluarea modelelor fundamentale
- ▣ Frecvențe, domenii, orizonturi diverse
- ▣ Metrici de calibrare și fiabilitate

Rezultate agregate pe benchmark-uri

Rezumat performanță cross-benchmark

- Clasamente agregate pe Monash, M3, M4, ETT, Weather, Electricity și Traffic
- Niciun model nu câștigă peste tot — **selecția modelului depinde de cazul de utilizare**
- FM sunt **cele mai impresionante în zero-shot**, egalând adesea bazele supervizate



Hartă termică: model (rânduri) $\times$ benchmark (coloane). Culoare = clasament relativ (verde = sus, roșu = jos).

TSA_ch11_benchmarks



Când să folosiți ce: Tabel decizional

Recomandări bazate pe scenarii

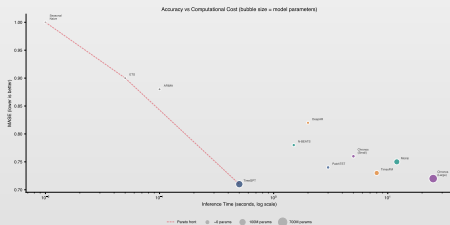
Scenariu	ARIMA	Chronos	TimesFM	Moirai	TimeGPT
Date limitate (<100 obs)	✓✓	✓	✓	~	✓
Timp real / latență mică	✓✓	~	✓	~	✓
Interpretabilitate necesară	✓✓	×	×	×	×
Dependențe multivariate	VAR	×	×	✓✓	~
Multe serii (10K+)	~	✓✓	✓✓	✓✓	✓
Fără expertiză ML	✓	✓	~	~	✓✓
Open source necesar	✓✓	✓✓	✓	✓✓	×
Proгноze probabilistice	✓	✓✓	~	✓✓	✓

✓✓ = excelent, ✓ = bun, ~ = acceptabil, × = potrivire slabă

Analiză cost computațional

Frontiera Pareto: Acuratețe vs. Cost

- ▣ Modelele fundamentale schimbă **calcul pentru comoditate**: nu necesită antrenare, dar inferența este mai grea
- ▣ “Cel mai bun” model se află pe frontiera Pareto a acurateții și costului



Considerente de cost

- ▣ **ARIMA**: CPU, milisecunde, gratuit
- ▣ **Chronos-Mini**: 1 GPU, ~0.1s/serie
- ▣ **Chronos-Large**: 1 GPU (A100), ~1s/serie
- ▣ **TimesFM**: 1 GPU, ~0.05s/serie (rapid)
- ▣ **TimeGPT**: API, \$0.01–\$0.10/apel
- ▣ **Pre-antrenare de la zero**: 100+ GPU-ore (Chronos-Large)

Limitări actuale

Cinci limitări cheie ale modelelor fundamentale pentru serii de timp

1. **Majoritatea modelelor sunt doar univariate:** Nu pot capta interacțiunile între variabile (ex. preț-volum, temperatură-cerere). Doar Moirai încearcă suport multivariat.
2. **Ferestre de context limitate (512–4096 token-uri):** Nu pot valorifica istorii foarte lungi. Pentru date zilnice cu sezonabilitate anuală, 512 pași $\approx$ doar 1.5 ani.
3. **Fără raționament causal:** Modelele învață corelații, nu cauzalitate. Nu pot răspunde la “ce se întâmplă dacă ratele dobânzii cresc cu 1%?” — niciun model structural subiacent.
4. **Natură cutie neagră:** Ponderile de atenție nu sunt ușor interpretabile. Dificil de explicat *de ce* a fost făcută o anumită prognoză — problematic pentru industrii reglementate.
5. **“Halucinație” în prognozele probabilistice:** Traectoriile generate pot prezenta tipare nerealiste (ex. prețuri negative, creștere explozivă) care încalcă constrângerile domeniului.

Atenuare

Post-procesare (tăiere, reconciliere), ansamblu cu modele clasice, calibrare conformală.

Considerații etice

Proveniența datelor

- Datele de pre-antrenare pot include serii de timp **private sau proprietare** (ex. date interne de vânzări colectate din API-uri)
- Lipsa transparenței: majoritatea modelelor nu dezvăluie complet compoziția datelor de antrenare
- LOTSA (Moirai) este un exemplu pozitiv: surse deschise, curate, documentate

Echitate și prejudecăți

- Pre-antrenarea se orientează către **domenii bogate în date** (energie, retail)
- Subreprezentate: sănătatea din țările în curs de dezvoltare, agricultura din Africa
- Antrenare părtinitoare → prognoze părtinitoare

Mediu și reglementare

- **Amprenta de carbon**: pre-antrenarea Chronos-Large $\approx 100+$ GPU-ore pe A100
- **EU AI Act**: sistemele AI cu risc ridicat necesită explicabilitate — modelele fundamentale pot să nu se califice
- **Responsabilitate**: cine este responsabil când o decizie bazată pe prognoză eșuează?

Direcții viitoare

Încotro se îndreaptă domeniul?

1. **Modele TS multimodale:** Combinarea seriilor de timp cu text (știri, rapoarte), imagini (satelit) și metadata.
2. **Prognoză agentică:** Agenți LLM care *selectează, rulează și evaluează* modele de prognoză autonom.
3. **Pre-antrenare continuă:** Actualizarea FM cu date noi fără re-antrenare completă. Critică pentru adaptarea la schimbări de distribuție.
4. **FM multivariate:** Depășirea universului univariat. Învățarea dinamicilor între variabile, structuri ierarhice și corelații spațiale.
5. **FM specifice domeniului:** FinanceFM, EnergyFM, HealthFM — mai mici, specializate, mai precise în domeniul lor.

Următorii 3–5 ani

Modelele fundamentale vor deveni probabil punctul de plecare implicit pentru prognoză, la fel cum embedding-urile pre-antrenate au devenit standard în NLP. Metodele clasice vor rămâne esențiale pentru interpretabilitate, inferență causală și regimuri cu date puține.

O perspectivă echilibrată

Când să folosiți modele clasice

- ▣ **Interpretabilitate necesară:** reglementare, cercetare științifică, analiză de politici
- ▣ **Inferență causală:** modele structurale (VAR, SVAR) pentru analize “ce-ar fi dacă”
- ▣ **Date puține:** <50 observații, structură sezonieră cunoscută a priori
- ▣ **Constrângeri computaționale:** dispozitive edge, sisteme real-time, fără GPU
- ▣ **Mandat de explicabilitate:** EU AI Act, MiFID II, Basel III

Când să folosiți modele fundamentale

- ▣ **Prototipare rapidă:** bază de referință puternică în minute, nu zile
- ▣ **Cross-domeniu:** același model pt. retail, energie, finanțe — fără expertiză necesară
- ▣ **Zero-shot / pornire la rece:** produs nou, piață nouă, fără date istorice
- ▣ **Scară:** 10,000+ serii — FM amortizează costul
- ▣ **Prognoze probabilistice:** cuantificare nativă a incertitudinii

Concluzia: Modelele clasice și fundamentale sunt complementare, nu concurente. Cei mai buni practicieni le folosesc pe amândouă.

Exercițiu IA: Selecția modelului fundațional

Prompt de testat în ChatGPT / Claude / Copilot

"Compare ARIMA, Prophet, and a foundation model (Chronos) for forecasting monthly airline passengers (the classic Box-Jenkins dataset, 144 observations). For each model: (1) fit the model, (2) forecast 24 months ahead, (3) plot actual vs predicted with 95% confidence intervals, (4) compute RMSE and MAPE. Which model would you recommend and why? Provide complete Python code."

Exercițiu:

1. Rulați prompt-ul într-un LLM la alegere și analizați critic răspunsul.
2. Gestionează codul corect **sezonalitatea multiplicativă** din datele de pasageri aerieni?
3. Cum gestionează Chronos sezonalitatea **fără a i se spune** că perioada este 12?
4. Sunt intervalele de încredere de la Chronos (bazate pe cuantile) comparabile cu cele ale ARIMA (parametrice)?
5. Discută LLM-ul **compromisul**: ARIMA este interpretabil, Chronos este zero-shot?

Avertisment: Codul generat de IA poate rula fără erori și arăta profesionist. *Aceasta nu înseamnă că este corect.* Validați întotdeauna față de benchmark-uri cunoscute.

Concluzii cheie

Ce am învățat

- ▣ **Transformerele** înlocuiesc recurența cu auto-atenția: paralele, scalabile, dependențe pe termen lung
- ▣ **Modelele fundamentale** sunt pre-antrenate pe miliarde de puncte temporale diverse și implementate zero-shot
- ▣ Strategiile de **tokenizare** (patching, cuantizare) fac punte între TS continue și intrările Transformer
- ▣ **Chronos** (T5, cuantizat), **TimesFM** (decoder, patched), **Lag-Llama** (LLaMA), **Moirai** (BERT, orice-variabilă) — filozofii de design distincte
- ▣ **Zero-shot** este surprinzător de puternic; **ajustarea fină** (LoRA) ajută cu date de domeniu

Recomandări practice

- ▣ Comparați întotdeauna cu **ARIMA/ETS** ca bază de referință clasică
- ▣ Folosiți **modele fundamentale** pentru prototipare rapidă și când expertiza de domeniu este limitată
- ▣ Modelele clasice și fundamentale sunt **complementare** — combinați-le prin ansambluri

Întrebarea 1

Întrebarea

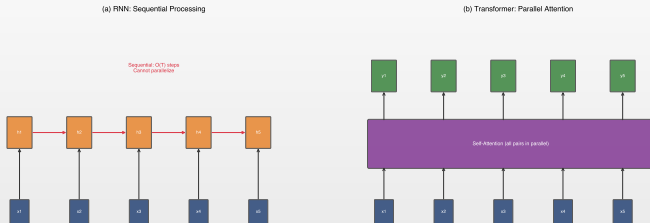
- ☐ Care este avantajul cheie al auto-atenției față de recurență în modelarea secvențelor?

Variante de răspuns

- (A) Auto-atenția folosește mai puțini parametri decât RNN-urile
- (B) Auto-atenția procesează toate pozițiile în paralel cu lungimea căii $O(1)$ între oricare două poziții
- (C) Auto-atenția nu necesită codificare pozițională
- (D) Auto-atenția este întotdeauna mai rapidă decât recurența indiferent de lungimea secvenței

Întrebarea 1: Răspuns

RNN vs Transformer: Processing Paradigm



Răspuns: (B)

- Auto-atenția conectează fiecare poziție cu fiecare altă poziție direct (lungimea căii $O(1)$), permițând paralelizare completă. RNN-urile necesită $O(T)$ pași secvențiali, cauzând gradienti care se estompează și antrenare lentă.

Întrebarea 2

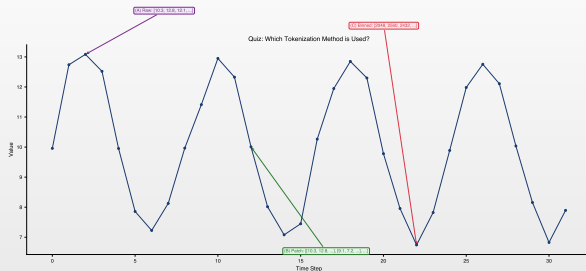
Întrebarea

- ▣ Ce strategie de tokenizare folosește Chronos pentru a converti seriile de timp continue în token-uri discrete?

Variante de răspuns

- (A) Patching: gruparea punctelor temporale consecutive în patch-uri
- (B) Scalare și cuantizare în bin-uri discrete folosind CDF-ul normal
- (C) Embedding liniar direct al valorilor brute
- (D) Byte-pair encoding (BPE) împrumutat din NLP

Întrebarea 2: Răspuns



Răspuns: (B)

- Chronos normalizează valorile (scalare prin medie), apoi le mapează la 4096 bin-uri prin $b_t = \lfloor (B - 1) \cdot \Phi(\tilde{x}_t) \rfloor$. Aceasta permite reutilizarea vocabularului T5 și a funcției de pierdere cross-entropy, transformând prognoza într-o problemă de modelare a limbajului peste token-uri cuantizate.

Întrebarea 3

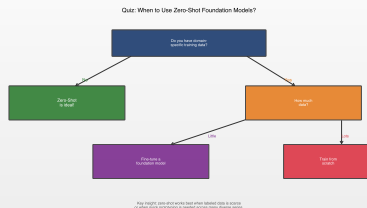
Întrebarea

- ☐ Când ar trebui să preferați inferența zero-shot în locul ajustării fine a unui model fundațional?

Variante de răspuns

- (A) Când aveți milioane de observații etichetate specifice domeniului
- (B) Când aveți nevoie de cea mai mare acuratețe posibilă pe un singur set de date
- (C) Când aveți date etichetate limitate, aveți nevoie de implementare rapidă sau vă confrunțați cu probleme de pornire la rece
- (D) Când resursele computaționale nu sunt o preocupare

Întrebarea 3: Răspuns



Răspuns: (C)

- Zero-shot excellează cu date limitate, pornire la rece și prototipare rapidă. Cunoștințele pre-antrenate ale modelului transferă tipare temporale din miliarde de serii diverse. Ajustarea fină este preferată când sunt disponibile suficiente date de domeniu și câștigurile marginale de acuratețe justifică costul.

Întrebarea 4

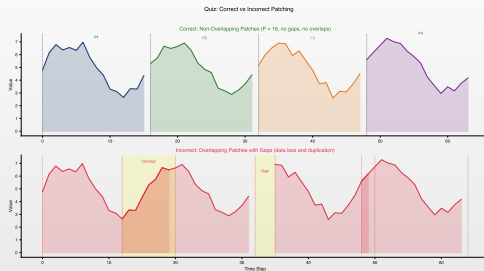
Întrebarea

▣ Ce este “patching” în contextul modelelor de serii de timp bazate pe Transformer și de ce ajută?

Variante de răspuns

- (A) Patching înseamnă completarea valorilor lipsă din seria de timp
- (B) Patching grupează P puncte consecutive într-un singur token, reducând complexitatea atenției de la $O(T^2)$ la $O((T/P)^2)$ captând în același timp tipare locale
- (C) Patching este o tehnică de augmentare a datelor care creează serii noi de antrenare
- (D) Patching se referă la actualizarea ponderilor modelului după implementare

Întrebarea 4: Răspuns



Răspuns: (B)

- Patching (PatchTST, 2023) grupează P puncte consecutive într-un singur token. Aceasta reduce lungimea efectivă a secvenței pentru atenție, captează tipare temporale locale în fiecare patch și permite modelului să gestioneze ferestre de context mult mai lungi.

Întrebarea 5

Întrebarea

- ☐ Trebuie să prognozați un set de date energetice multivariate cu 50 variabile corelate la frecvență orară. Care model fundațional este cel mai potrivit?

Variante de răspuns

- (A) Chronos — deoarece are cele mai multe dimensiuni de model din care să alegeți
- (B) TimeGPT — deoarece este cel mai ușor de folosit prin API
- (C) Moirai — deoarece este singurul model proiectat să gestioneze nativ orice număr de variabile și orice frecvență
- (D) Lag-Llama — deoarece folosește caracteristici lag care captează dependențele temporale

Întrebarea 5: Răspuns

Quiz: Model Selection by Scenario

Scenario	Data Size	Domain	Latency Requirement	Recommended Model
1. Retail demand forecasting (100M x 2 years)	Large (100M+ rows)	Retail	Low	Chronos (Highly flexible)
2. Manufacturing efficiency	Medium (50M rows)	Manufacturing	Medium	Chronos (Highly flexible)
3. High-frequency stock market	Very Large (100M+ rows)	Finance	Very Low	Real-time (Highly flexible)
4. Communication network	Small (10M rows)	Telecom	High	Chronos (Highly flexible)
5. IoT sensor network data	Medium (50M rows)	Industrial	Low	Chronos (Highly flexible)

Răspuns: (C)

- Moirai este singurul model fundațional proiectat pentru prognoze **orice-variabilă**. Folosește ID-uri de variabilă și patching flexibil pentru a gestiona 50 variabile corelate simultan. Chronos și Lag-Llama sunt univariate; ar procesa fiecare variabilă independent, ignorând corelațiile între variabile.

Bibliografie I

Lucrări fundamentale

- ▣ Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention Is All You Need, *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 5998–6008.
- ▣ Ansari, A.F., Stella, L., Turkmen, C., et al. (2024). Chronos: Learning the Language of Time Series, *arXiv preprint arXiv:2403.07815*.
- ▣ Das, A., Kong, W., Leber, A., et al. (2024). A Decoder-Only Foundation Model for Time-Series Forecasting, *Proceedings of ICML 2024*.

Foundation Models

- ▣ Garza, A., & Mergenthaler-Canseco, M. (2024). TimeGPT-1, *International Journal of Forecasting*, forthcoming.
- ▣ Rasul, K., Ashok, A., Williams, A.R., et al. (2024). Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting, *Proceedings of NeurIPS 2024*.

Bibliografie II

Arhitectură și pre-antrenare

- ▣ Woo, G., Liu, C., Kumar, A., et al. (2024). Unified Training of Universal Time Series Forecasting Transformers, *Proceedings of ICML 2024*.
- ▣ Nie, Y., Nguyen, N.H., Sinthong, P., & Kalagnanam, J. (2023). A Time Series is Worth 64 Words: Long-term Forecasting with Transformers, *Proceedings of ICLR 2023*.
- ▣ Bommasani, R., Hudson, D.A., Adeli, E., et al. (2021). On the Opportunities and Risks of Foundation Models, *arXiv preprint arXiv:2108.07258*.

Benchmark-uri și evaluare

- ▣ Godahewa, R., Bergmeir, C., Webb, G., et al. (2021). Monash Time Series Forecasting Archive, *Proceedings of NeurIPS 2021 Datasets and Benchmarks Track*.
- ▣ Hu, E.J., Shen, Y., Wallis, P., et al. (2022). LoRA: Low-Rank Adaptation of Large Language Models, *Proceedings of ICLR 2022*.

Bibliografie III

Referințe suplimentare

- Kaplan, J., McCandlish, S., Henighan, T., et al. (2020). Scaling Laws for Neural Language Models, *arXiv preprint arXiv:2001.08361*.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2018). The M4 Competition: 100,000 Time Series and 61 Forecasting Methods, *International Journal of Forecasting*, 36(1), 54–74.
- Petropoulos, F., et al. (2022). Forecasting: Theory and Practice, *International Journal of Forecasting*, 38(3), 845–1054.

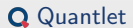
Resurse online și cod

- **Quantlet:** <https://quantlet.com> – Platformă de cod pentru metode cantitative
- **Quantinar:** <https://quantinar.com> – Platformă de învățare pentru metode cantitative
- **GitHub TSA:** https://github.com/danpele/TSA/tree/main/TSA_ch11 – Cod Python pentru acest capitol

Vă Mulțumim!

Întrebări?

Materialele cursului sunt disponibile la: <https://danpele.github.io/Time-Series-Analysis/>



Quantlet



Quantinar