

Neural Networks and Deep Learning

Lecture 5: Recurrent networks: RNN, LSTM and GRU for sequences and financial time series





Prof. dr. Daniel Traian PELE

Bucharest University of Economic Studies (ASE), Department of Statistics and Econometrics
Romanian Academy, Institute for Economic Forecasting (IPE)
IDA Institute Digital Assets ▪ AI4EFin ▪ MSCA Digital Finance

Learning objectives

- Write a recurrent neural network (RNN) as equations, unroll it in time and count its parameters.
- Derive backpropagation through time (BPTT) and explain why gradients vanish or explode.
- Read the long short-term memory (LSTM) and gated recurrent unit (GRU) cells gate by gate.
- Forecast volatility, returns and electricity load with a leakage-free protocol, after the benchmarks: heterogeneous autoregressive (HAR), generalised autoregressive conditional heteroskedasticity (GARCH), autoregressive (AR) and autoregressive with exogenous inputs (ARX) models.
- Explain why a multilayer perceptron (MLP) on a window of past values is not enough.
- Decide with a forecast-comparison test when a recurrent network is worth its cost.

Abbreviations

AI	artificial intelligence
AR	autoregressive model
ARX	autoregressive model with exogenous inputs
ASE	Bucharest University of Economic Studies
BiLSTM	bidirectional LSTM
BPTT	backpropagation through time
BVB	Bucharest Stock Exchange
CEE	Central and Eastern Europe
CNN	convolutional neural network
CPU	central processing unit
DM	Diebold–Mariano test
ENTSO-E	European Network of Transmission System Operators for Electricity
EODHD	End of Day Historical Data (data provider)
ES	expected shortfall
GARCH	generalised autoregressive conditional heteroskedasticity
GK	Garman–Klass range estimator of variance
GRU	gated recurrent unit
GW	gigawatt
HAR	heterogeneous autoregressive model
IPE	Institute for Economic Forecasting, Romanian Academy
LSTM	long short-term memory
MAE	mean absolute error
MAPE	mean absolute percentage error
MLP	multilayer perceptron
MSE	mean squared error
MW	megawatt
MWh	megawatt-hour
RNN	recurrent neural network
TLV	Banca Transilvania (Bucharest Stock Exchange ticker)
VaR	value-at-risk

The data: three real sequences

Series	Source	Period	Task
S&P 500 daily open, high, low, close	Yahoo Finance	4 Jan 2005 – 30 Oct 2026	volatility, returns
Banca Transilvania (TLV), adjusted close	End of Day Historical Data (EODHD), Bucharest Stock Exchange (BVB)	3 Jan 2008 – 30 Oct 2026	returns
Romanian hourly load	European Network of Transmission System Operators for Electricity (ENTSO-E), via energy-charts.info	1 Jan 2016 – 31 Oct 2026	day-ahead load

- Time-ordered splits: train to 31 Dec 2016, validation to 31 Dec 2019, test from 1 Jan 2020 (load: train to 31 Dec 2022, validation 31 Dec 2023).
- Scalers, early stopping and penalties are fitted on the training or validation years only; the test years are used once.
- Networks trained on the central processing unit (CPU), deterministic for a fixed seed (20261108 + seed index).

(data/SOURCES.md)

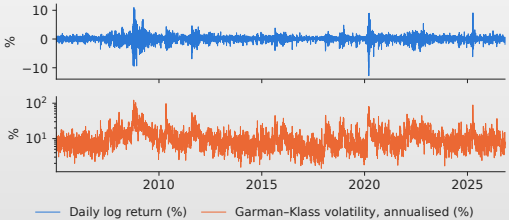
Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

Returns and volatility clustering

- **In one sentence:** the size of the move today says a lot about the size tomorrow; its sign says almost nothing.

$$r_t = 100 \ln \frac{P_t}{P_{t-1}}, \qquad \hat{\sigma}^2_{\mathrm{GK},t} = \tfrac{1}{2}\Big(\ln \frac{H_t}{L_t}\Big)^2 - (2\ln 2 - 1)\Big(\ln \frac{C_t}{O_t}\Big)^2$$



Source: Yahoo Finance, daily quotes. Data: 4 Jan 2005 to 30 Oct 2026, trading days.

- Garman–Klass (GK) range estimator from open O_t, high H_t, low L_t, close C_t
  - Annualised volatility median 8.77 %, peak 122 % on 10 Oct 2008.

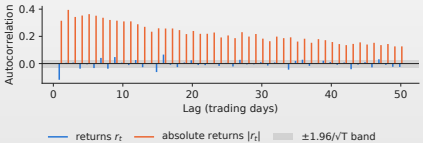
(Garman & Klass, 1980; own computation)

NN_L5_sp500_vol

Autocorrelation of returns and of absolute returns

- **In one sentence:** autocorrelations of $|r_t|$ stay positive for months; those of r_t are within the noise band.

$$\hat{\rho}(k) = \frac{\sum_{t=k+1}^{T}(x_t - \bar{x})(x_{t-k} - \bar{x})}{\sum_{t=1}^{T}(x_t - \bar{x})^2}$$



Source: Yahoo Finance, daily quotes. Data: 4 Jan 2005 to 30 Oct 2026, trading days.

- $\hat{\rho}_r(1) = -0.119$ but $\hat{\rho}_{|r|}(1) = 0.315$ and $\hat{\rho}_{|r|}(50) = 0.126$
  - Band ± 0.026
  - Significant lags out of 50: 15 for r_t, 50 for $|r_t|$.

- **In practice:** a model with memory is worth trying for risk (volatility, value-at-risk (VaR)), much less for the direction of the price.

(own computation)

NN_L5_acf_abs

Limited memory of a fixed-window MLP

⊡ **In one sentence:** an MLP on the last W observations has separate weights for every lag and a hard memory limit.

$$\hat{y}_{t+1} = g\big(\mathbf{W}[\mathbf{x}_{t-W+1}; \ldots; \mathbf{x}_t] + \mathbf{b}\big), \qquad \mathbf{W} \in \mathbb{R}^{m \times Wd}$$

⊡ **where**

- $\mathbf{x}_t \in \mathbb{R}^d$ the inputs of day t
- W window length
- m hidden units
- Parameters grow linearly with W

⊡ **What we want instead**

- A state $\mathbf{h}_t$ that summarises the past
- The same update rule at every step (weight sharing in time, like a convolutional neural network (CNN) in space)
- Any sequence length

⊡ **In practice:** in production the state is updated with each new observation, without recomputing the whole history.

(Bishop & Bishop, 2024, ch. 12)

Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

The vanilla RNN: one state, updated by the same rule

⊡ **In one sentence:** the new state mixes the old state and the new input through a nonlinearity.

$$\mathbf{h}_t = \tanh(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h), \qquad \hat{y}_t = \mathbf{W}_{hy}\mathbf{h}_t + b_y$$

⊡ **where**

- $\mathbf{h}_t \in \mathbb{R}^{n_h}$ hidden state, $\mathbf{h}_0 = \mathbf{0}$
- $\mathbf{W}_{xh} \in \mathbb{R}^{n_h \times d}$, $\mathbf{W}_{hh} \in \mathbb{R}^{n_h \times n_h}$ shared across t
- Parameters $n_h(d + n_h + 1) + n_h + 1$, independent of the length T

⊡ Unrolled in time it is a deep network of depth T with tied weights.

⊡ Our volatility model: $d = 3$ inputs (log variance, return, absolute return), $n_h = 32$, window $T = 22$ trading days.

⊡ Many-to-one: only $\hat{y}_T$ (tomorrow's log variance) enters the loss.

⊡ **In practice:** the hidden state is a learned, nonlinear analogue of the conditional variance of a GARCH model.

(Rumelhart et al., 1986; Bishop & Bishop, 2024, sec. 12.2)

Backpropagation through time: a sum over paths

- **In one sentence:** the gradient of a shared weight adds its contribution at every step, each passing through a product of Jacobians.

$$\frac{\partial \mathcal{L}_T}{\partial \mathbf{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_T}{\partial \mathbf{h}_T} \Big(\prod_{s=t+1}^{T} \mathbf{J}_s \Big) \frac{\partial^+ \mathbf{h}_t}{\partial \mathbf{W}_{hh}}$$

$$\mathbf{J}_s = \frac{\partial \mathbf{h}_s}{\partial \mathbf{h}_{s-1}} = \mathrm{diag}(1 - \mathbf{h}_s^2)\mathbf{W}_{hh}$$

- **where**
  - ∂^+ the immediate derivative, with $\mathbf{h}_{t-1}$ held fixed
  - $\mathbf{J}_s$ Jacobian of one step
  - The derivation is in the appendix

- Cost: one backward pass through all T steps, memory $O(Tn_h)$.
- Truncated BPTT: back-propagate only $K < T$ steps
  - Cheaper, but no gradient signal beyond K.

- **In practice:** the product of $T - t$ Jacobians decides what the network can learn from far in the past.

(Werbos, 1990)

Vanishing and exploding gradients

- **In one sentence:** a product of many matrices shrinks or grows geometrically with the distance in time.

$$\left\| \prod_{s=t+1}^{T} \mathbf{J}_s \right\| \leq \left(\gamma\, \sigma_{\max}(\mathbf{W}_{hh})\right)^{T-t}, \qquad \gamma = \max |\tanh'| \leq 1$$

- **where**
  - $\sigma_{\max}$ largest singular value
  - If $\gamma\sigma_{\max} < 1$ the bound vanishes geometrically (sufficient condition)
  - A largest eigenvalue above 1 is necessary for explosion

- **Exploding**: clip the gradient norm
  - $\mathbf{g} \leftarrow \mathbf{g} \cdot \min(1,\, c/\|\mathbf{g}\|)$; we use $c = 1$ (chosen a priori)
- **Vanishing**: change the architecture
  - An additive path through time: the LSTM and GRU cells, next section

- **In practice:** a vanilla RNN in practice remembers a few dozen steps: enough for daily volatility, not for weekly patterns in hourly data.

(Bengio et al., 1994; Pascanu et al., 2013)

Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

The LSTM cell and its gates

- **In one sentence:** a separate cell state $\mathbf{c}_t$ is updated additively; three gates decide what to forget, write and show.

$$\mathbf{f}_t = \sigma(\mathbf{W}_f[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_f), \qquad \mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_i),$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_o), \qquad \tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_c),$$
$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t, \qquad \mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

- **where**
  - σ logistic function, values in $(0, 1)$
  - $\odot$ element-wise product
  - $\mathbf{f}, \mathbf{i}, \mathbf{o}$ forget, input, output gates
  - Parameters $4n_h(n_h + d + 1)$

- Forget gate near 1: the cell keeps its memory
  - Near 0: it resets.
- Input gate: how much of the candidate $\tilde{\mathbf{c}}_t$ to write.
- Output gate: how much of the memory is exposed in $\mathbf{h}_t$.

- **In practice:** the gates are learned switches
  - On market data they react to shocks (two frames below).

(Hochreiter & Schmidhuber, 1997)

Why the cell state carries gradients far back

- **In one sentence:** along the cell state the Jacobian is a diagonal of forget gates, not a weight matrix squashed by $\tanh'$.

$$\left.\frac{\partial \mathbf{c}_t}{\partial \mathbf{c}_{t-1}}\right|_{\text{direct}} = \operatorname{diag}(\mathbf{f}_t), \qquad \prod_{s=t+1}^{T} \operatorname{diag}(\mathbf{f}_s) \approx \mathbf{I} \text{ if } \mathbf{f}_s \approx \mathbf{1}$$

- **where**
  - The other paths go through $\mathbf{h}_{t-1}$ and the gates
  - The direct path is what keeps long-range gradients alive

- Forget-gate bias initialised positive (e.g. 1) makes $\mathbf{f}_t \approx \sigma(1) = 0.73$ at the start: memory by default.
- On our volatility test period the mean forget gate is 0.547 and the mean input gate 0.544.
- The additive update is the same idea as the residual connection of Lecture 4.

- **In practice:** choose an LSTM or GRU when the useful past is long (a week of hours), a vanilla RNN only for short memories.

(Hochreiter & Schmidhuber, 1997)

The GRU: two gates and no separate cell

- **In one sentence:** an update gate interpolates between the old state and a candidate; a reset gate decides how much past enters the candidate.

$$\mathbf{z}_t = \sigma(\mathbf{W}_z[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_z), \qquad \mathbf{r}_t = \sigma(\mathbf{W}_r[\mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_r),$$
$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_h[\mathbf{r}_t \odot \mathbf{h}_{t-1}; \mathbf{x}_t] + \mathbf{b}_h), \quad \mathbf{h}_t = (1 - \mathbf{z}_t) \odot \mathbf{h}_{t-1} + \mathbf{z}_t \odot \tilde{\mathbf{h}}_t$$

- **where**
  - $\mathbf{z}_t$ update gate
  - $\mathbf{r}_t$ reset gate
  - Parameters $3n_h(n_h + d + 1)$, three quarters of an LSTM with the same n_h

- Same additive path: $\partial \mathbf{h}_t / \partial \mathbf{h}_{t-1} \supset \mathrm{diag}(1 - \mathbf{z}_t)$.
- LSTM vs GRU: no general winner
  - Compare both on validation data, as we do below.

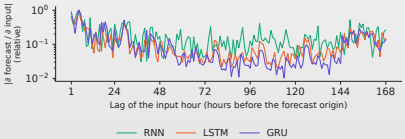
- **In practice:** fewer parameters make the GRU slightly faster to train and less data-hungry.

(Cho et al., 2014)

Input sensitivity by lag: the networks rely on the previous day and the previous week

- **In one sentence:** sensitivity of the output of the trained load networks to each of the 168 input hours, averaged over 200 test days.

$$S(k) = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{\partial \hat{y}_{i,12}}{\partial x_{i,\,T+1-k}} \right|$$



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 200 test days of 2024; models with seed 20261108; forecast for noon (hour 12, universal time).

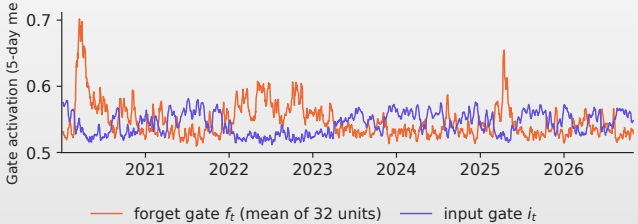
- Reading 7 daily profiles instead of 168 hours shortens the gradient path to 7 steps
  - At a lag of one week the sensitivity, relative to the last day's maximum, is RNN 0.261, LSTM 0.131, GRU 0.140.

(own computation)

NN_L5_grad_lag

LSTM gates react to market stress

- **In one sentence:** gate activations of the trained volatility LSTM on the test years 2020–2026 (5-day means).



Source: Yahoo Finance, daily quotes. Data: test period 2020-2026, LSTM seed 20261108, last step of each 22-day window.

- Correlation with today's log variance: forget gate 0.743, input gate −0.616.

- **In practice:** gates are a diagnostic, not an explanation: use them to see when the model changes regime, then check with the data.

(own computation)

NN_L5_lstm_gates

Sequence architectures and look-ahead leakage

- **Many-to-one**
  - A window in, one value out: tomorrow's volatility
- **Many-to-many (vector output)**
  - A week of daily profiles in, 24 hours of tomorrow out: our load model
- **Encoder–decoder**
  - One network reads, another writes a sequence step by step
- **Stacked**
  - The states of layer 1 are the inputs of layer 2
- **Bidirectional (BiLSTM)**
  - Reads the window forward and backward
  - Fine for classifying a finished text or sequence
  - A leak in forecasting if the backward pass sees the future

- **In practice:** in forecasting every input must be known at the forecast origin: check this before any architecture choice.

(Bishop & Bishop, 2024, ch. 12)

Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

Windows, scaling and splits without leakage

- **In one sentence:** every number the model sees must be computable at the forecast origin t.

$$\mathcal{D} = \{(\mathbf{x}_{t-W+1:t},\, y_{t+1})\}, \qquad \tilde{x} = \frac{x - \bar{x}_{\text{train}}}{s_{\text{train}}}$$

- **where**
  - W window
  - $\bar{x}_{\text{train}}, s_{\text{train}}$ mean and standard deviation of the training years only

- S&P 500: 3 020 training, 754 validation and 1 717 test days, in this order.
- Early stopping: stop when the validation loss has not improved for 6 epochs
  - Keep the best weights.
- Seeds: 5 per network (volatility, returns), 3 for load
  - We report the spread and the mean forecast.

- **In practice:** a random shuffle of days before splitting puts 2020 in the training set and makes any model look good on 2020.

(own computation)

Experiment yourself: §2

Benchmark models: naive, HAR and GARCH

- **In one sentence:** a network must beat models that econometrics has refined for decades.

$$\text{HAR: } y_{t+1} = \beta_0 + \beta_d y_t + \beta_w \bar{y}_{t-4:t} + \beta_m \bar{y}_{t-21:t}$$

$$\text{GARCH(1,1): } \sigma^2_{t+1} = \omega + \alpha\, \varepsilon_t^2 + \beta\, \sigma_t^2, \quad \varepsilon_t = r_t - \mu$$

- HAR estimates: $\hat{\beta}_d = 0.192$, $\hat{\beta}_w = 0.439$, $\hat{\beta}_m = 0.293$.
- GARCH: $\hat{\alpha} = 0.117$, $\hat{\beta} = 0.873$, persistence 0.990, $\hat{\nu} = 5.78$.

- **where**
  - $y_t = \ln \hat{\sigma}^2_{\mathrm{GK},t}$
  - $\bar{y}$ means over a week and a month
  - GARCH with Student-t errors, mapped to y by a linear regression on the training years
  - Naive: $\hat{y}_{t+1} = y_t$

- **In practice:** HAR is a linear model with three averages
  - If an LSTM cannot beat it, the LSTM is not worth maintaining.

(Corsi, 2009; Bollerslev, 1986)

The Diebold–Mariano test of equal predictive accuracy

- **In one sentence:** compare the losses of two forecasts day by day and test whether their mean difference is zero.

$$d_t = L(e_{1,t}) - L(e_{2,t}), \qquad \mathrm{DM} = \frac{\bar{d}}{\sqrt{\hat{\sigma}^2_{\mathrm{NW}}/n}} \overset{H_0}{\sim} \mathcal{N}(0,1)$$

- **where**
  - L squared error (volatility) or absolute error (load)
  - $\hat{\sigma}^2_{\mathrm{NW}}$ Newey–West long-run variance of d_t
  - $\mathrm{DM} < 0$: model 1 better

- The Diebold–Mariano (DM) test accounts for autocorrelated loss differences, which forecast errors always have.
- Seeds measure the training noise
  - DM measures the sampling noise of the test period.

- **In practice:** report a model as better only with a DM p-value, and say over which period.

(Diebold & Mariano, 1995)

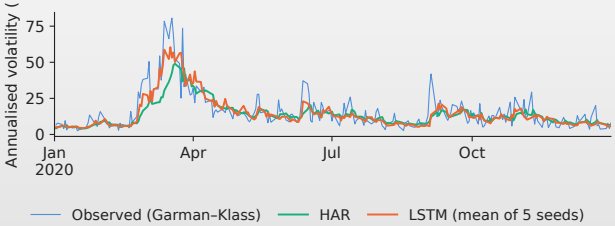
Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

In the 2020 crash the LSTM adapted faster than HAR

- **In one sentence:** one-day-ahead forecasts of annualised volatility in the first test year.

$$\hat{\sigma}^{\text{ann}}_{t+1} = 100\sqrt{252\,\exp(\hat{y}_{t+1})}$$



Source: Yahoo Finance, daily quotes. Data: test year 2020 (one-day-ahead forecasts), trading days.

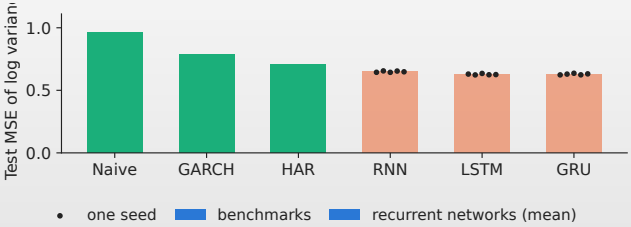
- Crash window (20 Feb – 30 Apr 2020), mean squared error (MSE) of $\hat{y}$: naive 0.705, GARCH 1.032, HAR 1.014, LSTM 0.521.

(own computation)

NN_L5_vol_forecast

Recurrent networks beat HAR and GARCH on volatility

- **In one sentence:** test MSE of the log variance over 2020–2026; dots are single seeds.



Source: Yahoo Finance, daily quotes. Data: test 2 Jan 2020 to 31 Oct 2026; 5 seeds per network.

- MSE: naive 0.966, GARCH 0.786, HAR 0.706
  - RNN 0.649, LSTM 0.628, GRU 0.629 (seed means).

(own computation)

NN_L5_vol_mse

Volatility results with tests

Model	MSE	R^2	Seeds
Naive	0.966	0.259	–
GARCH(1,1)-t	0.786	0.397	–
HAR	0.706	0.458	–
RNN (mean forecast)	0.643	0.507	5
LSTM (mean forecast)	0.625	0.521	5
GRU (mean forecast)	0.624	0.521	5

- $R^2 = 1 - \mathrm{MSE}/\widehat{\mathrm{Var}}(y)$ on the 1 717 test days.

- LSTM vs HAR: MSE change $-11.6\,\%$
  - DM $= -5.52$, $p < 0.001$.
- LSTM vs GARCH: DM $= -6.80$, $p < 0.001$.
- LSTM vs naive: MSE change $-35.3\,\%$.
- Best network by mean forecast: GRU
  - Seed spread of the LSTM: 0.623 to 0.635.

- **In practice:** the gain is significant on this test period
  - Before replacing HAR in a VaR model, check it on another market and sub-period and keep HAR as the fallback.

(own computation)

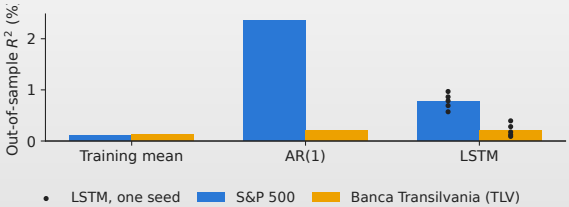
Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

Return forecasts explain almost none of the out-of-sample variance

- **In one sentence:** next-day returns of the S&P 500 and of Banca Transilvania (TLV) from the last 22 returns.

$$R^2_{\text{OOS}} = 1 - \frac{\sum_{t \in \text{test}} (r_{t+1} - \hat{r}_{t+1})^2}{\sum_{t \in \text{test}} r^2_{t+1}}$$



Source: Yahoo Finance; EODHD. Data: test 2020 to Oct 2026; S&P 500 (Yahoo Finance) and TLV (EODHD, Bucharest Stock Exchange).

- S&P 500: AR(1) 2.366 %, LSTM 0.776 % (seeds 0.572 to 0.969)
  - TLV: AR(1) 0.201 %, LSTM 0.213 %.

(own computation)

NN_L5_returns_r2

Low predictability of returns

- **In one sentence:** in a liquid market the predictable part of returns is tiny compared with their noise.

$$r_{t+1} = \mu_t + \varepsilon_{t+1}, \qquad \frac{\mathrm{Var}(\mu_t)}{\mathrm{Var}(r_{t+1})} \approx 0$$

- **where**
  - μ_t the conditional mean a model can learn
  - ε_{t+1} news that arrives after t

- LSTM sign accuracy on the S&P 500 test: 52.1 %, against 54.0 % up days (always "up").
- AR(1) slope estimated on the training years: -0.102 (S&P 500), 0.015 (TLV).
- Flexible models fit noise in training
  - Early stopping limits the damage, it cannot create a signal.

- **In practice:** a vendor's back-test of an "artificial intelligence (AI) return predictor" must pass this test on data it has never seen, after costs.

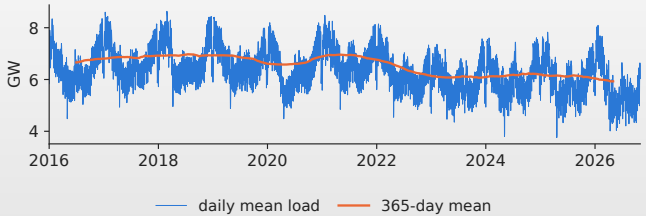
(own computation)

Outline

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

Load has daily, weekly and yearly cycles

- **In one sentence:** a sequence with long, regular memory: a natural case for recurrent networks.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 1 Jan 2016 to 31 Oct 2026, hourly, mean MW per day.

- Hourly load from 1 Jan 2016 to 31 Oct 2026, mean 6 510 megawatt (MW)
  - 129 missing hours stay missing (those days are skipped).

(own computation)

NN_L5_load_series

The day-ahead task and its benchmarks

- **In one sentence:** at the end of day d forecast the 24 hourly loads of day $d+1$ from the last week.

$$\hat{\mathbf{y}}_{d+1} = \mathbf{y}_{d-6} + \mathbf{W}_o[\mathbf{h}_7; \mathbf{s}_{d+1}] + \mathbf{b}_o \in \mathbb{R}^{24}$$

$$\text{ARX: } \hat{\mathbf{y}}_{d+1} = \mathbf{y}_{d-6} + \mathbf{B}\,\mathbf{x}_{d,1:168} + \grave{}\mathbf{s}_{d+1}$$

- 2 186 training, 365 validation, 1035 test days
  - 3 seeds per network.

- **where**
  - $\mathbf{y}_{d-6}$ seasonal naive forecast (the models learn its correction)
  - $\mathbf{h}_7$ last state after the 7 past days, one step per day (24 hourly loads + weekday)
  - $\mathbf{s}_{d+1}$ weekday and season of day $d+1$
  - ARX by ridge regression (penalty 1, chosen on the validation year)

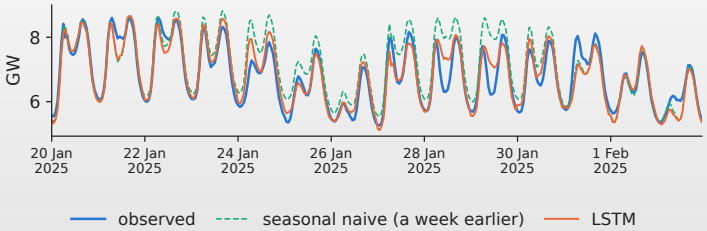
- **In practice:** a supplier submits the 24 hourly volumes before noon
  - Every MW of error is bought or sold at the imbalance price.

(own computation)

Experiment yourself: §5

Two winter test weeks: forecasts against the observed load

- **In one sentence:** hourly load in two winter test weeks, observed against two forecasts.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 14 test days from 20 Jan 2025 to 2 Feb 2025.

- The seasonal naive copies last week, including its level and its holidays
  - The networks learn only a correction of it.

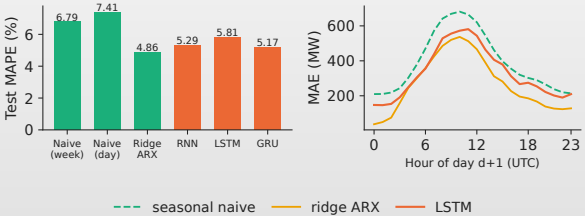
(own computation)

NN_L5_load_fortnight

On load, the linear ARX beats the recurrent networks

- **In one sentence:** left: mean absolute percentage error (MAPE) over all test hours; right: mean absolute error (MAE) by hour of day.

$$\mathrm{MAPE} = \frac{100}{24n} \sum_{d} \sum_{h=1}^{24} \frac{|y_{d,h} - \hat{y}_{d,h}|}{y_{d,h}}$$



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: 1 Jan 2024 to 31 Oct 2026 (test days).

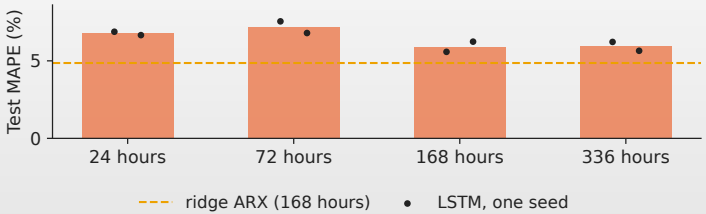
- MAPE (%): seasonal naive 6.79, naive 7.41, ridge ARX 4.86, RNN 5.29, LSTM 5.81, GRU 5.17.

(own computation)

NN_L5_load_errors

Longer windows help the LSTM, up to a week

⊡ **In one sentence:** LSTM test error as a function of the input window, same protocol otherwise.



Source: energy-charts.info (Fraunhofer ISE), from exchange and ENTSO-E data. Data: test days 2024-2026; LSTM with 2 seeds per window.

⊡ MAPE (%): 24 hours 6.77, 72 hours 7.17, 168 hours 5.91, 336 hours 5.93.

(own computation)

NN_L5_load_window

Economic value of a more accurate load forecast

- **Significance**
  - Best network by mean forecast: GRU, MAPE $+6.4\,\%$ relative to ridge ARX: with 2 186 training days the linear model wins
  - DM on daily MAE, LSTM vs ridge ARX: 9.80, $p < 0.001$
- **In megawatts**
  - Mean absolute error per hour: seasonal naive 396 MW, ridge ARX 269 MW, LSTM 331 MW
  - LSTM instead of the seasonal naive: about 1 560 megawatt-hours (MWh) less absolute error per day, at national scale
- **Cost**
  - Training of all load models and the window study: 67 seconds on a laptop CPU

- **In practice:** multiply the MWh saved by the imbalance price spread and compare with the cost of the model: here the ridge ARX saves the most for the least
  - Weather inputs would add more (Lecture 13).

(own computation)

Quiz 1: parameters

- How many parameters has an LSTM layer with $d = 3$ inputs and $n_h = 32$ units?

Answer

- $4n_h(n_h + d + 1) = 4 \cdot 32 \cdot 36 = 4608$ (PyTorch keeps two bias vectors, so it reports $4608 + 128 = 4736$).

Quiz 2: gradients

- Why does a vanilla RNN struggle with the weekly pattern of hourly load?

Answer

- Over 168 hourly steps the gradient is a product of 168 Jacobians and vanishes
  - Reading 7 daily profiles shortens the path to 7 steps, which is how our load models are built.

Quiz 3: a vendor's claim

- A vendor shows an LSTM with 60 % sign accuracy on S&P 500 returns, trained on shuffled days
  - What do you ask?

Answer

- A time-ordered test on unseen years, against "always up" (54.0 %) and after costs
  - Our honest LSTM reaches 52.1 %.

Quiz 4: HAR or LSTM

- The LSTM changes the volatility MSE by $-11.6\,\%$ against HAR, with DM $= -5.52$
  - Does it replace HAR in the VaR system?

Answer

- The gain is significant here, so the LSTM is a serious candidate
  - Switch only after it holds on another market and sub-period, with monitoring and HAR as the fallback.

Quiz 5: bidirectional

- Can you use a bidirectional LSTM to forecast tomorrow's load from a window that ends today?

Answer

- Yes, if the window ends at the forecast origin: the backward pass then sees only the past
  - It leaks only if the window contains hours after the origin.

Wrap-up

- Sequences in finance and energy
- The recurrent network and backpropagation through time
- Gated cells: LSTM and GRU
- A protocol for financial time series
- Case study A: S&P 500 volatility
- Case study B: forecasting returns with neural networks
- Case study C: Romanian electricity load

- An RNN shares weights in time
  - BPTT multiplies Jacobians, so gradients vanish or explode
  - LSTM and GRU add an additive memory path.
- On real data: the networks beat HAR on volatility, return forecasts explain almost nothing, and a linear ARX wins on hourly load.
  - Next: text (Lecture 6).

Research from our group: sequences and risk

- **Can foundation models manage risk?** (Jheng, Tak, Găman, Mazurencu-Marinescu-Pele & Pele, 2026, Economic Computation and Economic Cybernetics Studies and Research)
  - Pre-trained time-series models used without fine-tuning to forecast VaR and expected shortfall (ES) for Central and Eastern European (CEE) markets
  - A conformal calibration step makes the tail forecasts reliable; the transformer successors of the LSTM come in Lectures 7 and 13
- **Early warning of "zombie" crypto-assets** (Będowska-Sójka, Wójcik & Pele, 2026, North American Journal of Economics and Finance)
  - A machine-learning early-warning system for crypto-assets that lose their market activity
- **BitMood** (Conda, Găman, Bâg, Mazurencu-Marinescu-Pele, Pele & Härdle, 2026, Digital Finance)
  - Facebook sentiment and Bitcoin volume: linear regression and a bidirectional LSTM reach comparable test performance, so model complexity does not guarantee better forecasts
- Findings as stated in the papers; numbers on the other slides come from our own computations.

(Bucharest University of Economic Studies; Romanian Academy, Institute for Economic Forecasting)

Further reading and resources

- **Bishop & Bishop (2024)**, ch. 12: sequence models, from RNNs to transformers.
- **Pascanu, Mikolov & Bengio (2013)**: the clearest analysis of vanishing and exploding gradients.
- **Corsi (2009)**: the HAR model, the benchmark every volatility network must face.
- **This course**: code, data, notebooks — github.com/danpele/NEURAL_BIZ.
- **Experiment yourself**: the notebook reproduces the charts of this lecture — notebooks/.
- **Seminar 5**: a GRU for the volatility of another index, against HAR, with a DM test.

References (1/2)

- Bengio, Y., Simard, P. & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2), 157–166. doi:10.1109/72.279181
- Bishop, C. M. & Bishop, H. (2024). *Deep Learning: Foundations and Concepts*, ch. 12. Springer. doi:10.1007/978-3-031-45468-4
- Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3), 307–327. doi:10.1016/0304-4076(86)90063-1
- Będowska-Sójka, B., Wójcik, P. & Pele, D. T. (2026). Early warning systems for cryptocurrency markets: Predicting 'zombie' assets using machine learning. *North American Journal of Economics and Finance*, 81, 102543. doi:10.1016/j.najef.2025.102543
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H. & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *EMNLP*, 1724–1734. doi:10.3115/v1/D14-1179
- Conda, A. I., Găman, Ş., Bâg, R. C., Mazurencu-Marinescu-Pele, M., Pele, D. T. & Härdle, W. K. (2026). BitMood: AI analysis of Bitcoin trends via Facebook emotions. *Digital Finance*, 8(3), 42. doi:10.1007/s42521-026-00205-3
- Corsi, F. (2009). A simple approximate long-memory model of realized volatility. *Journal of Financial Econometrics*, 7(2), 174–196. doi:10.1093/jjfinec/nbp001
- Diebold, F. X. & Mariano, R. S. (1995). Comparing predictive accuracy. *Journal of Business & Economic Statistics*, 13(3), 253–263. doi:10.1080/07350015.1995.10524599

References (2/2)

- Garman, M. B. & Klass, M. J. (1980). On the estimation of security price volatilities from historical data. *Journal of Business*, 53(1), 67–78. doi:10.1086/296072
- Hochreiter, S. & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780. doi:10.1162/neco.1997.9.8.1735
- Jheng, S.-L., Tak, R., Găman, Ș., Mazurencu-Marinescu-Pele, M. & Pele, D. T. (2026). Can foundation models manage risk? Zero-shot VaR and ES forecasting with conformal calibration in CEE markets. *Economic Computation and Economic Cybernetics Studies and Research*, 60(2), 267–290. doi:10.24818/18423264/60.2.26.14
- Pascanu, R., Mikolov, T. & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. *ICML*, PMLR 28(3), 1310–1318. arXiv:1211.5063
- Rumelhart, D. E., Hinton, G. E. & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536. doi:10.1038/323533a0
- Werbos, P. J. (1990). Backpropagation through time: What it does and how to do it. *Proceedings of the IEEE*, 78(10), 1550–1560. doi:10.1109/5.58337

Appendix: BPTT for the vanilla RNN, step by step

- Pre-activation $\mathbf{a}_t = \mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h$, state $\mathbf{h}_t = \tanh(\mathbf{a}_t)$, loss $\mathcal{L} = \sum_t \ell_t(\hat{y}_t, y_t)$.
- Backward recursion: $\boldsymbol{\delta}_t \equiv \partial\mathcal{L}/\partial\mathbf{h}_t = \mathbf{W}_{hy}^\top \partial\ell_t/\partial\hat{y}_t + \mathbf{W}_{hh}^\top\left[(1-\mathbf{h}_{t+1}^2)\odot\boldsymbol{\delta}_{t+1}\right]$, with $\boldsymbol{\delta}_{T+1} = \mathbf{0}$.
- Weight gradients: $\partial\mathcal{L}/\partial\mathbf{W}_{hh} = \sum_t\left[(1-\mathbf{h}_t^2)\odot\boldsymbol{\delta}_t\right]\mathbf{h}_{t-1}^\top$, $\partial\mathcal{L}/\partial\mathbf{W}_{xh} = \sum_t\left[(1-\mathbf{h}_t^2)\odot\boldsymbol{\delta}_t\right]\mathbf{x}_t^\top$.
- Unrolling the recursion gives the product of Jacobians of the main text
  - Many-to-one models have $\partial\ell_t/\partial\hat{y}_t = 0$ for $t < T$.

Appendix: the models in PyTorch

```
class RecNet(nn.Module):
    def __init__(self, cell, n_in, hidden=32, n_out=1, n_static=0):
        super().__init__()
        self.rnn = {'rnn': nn.RNN, 'lstm': nn.LSTM, 'gru': nn.GRU}[cell](n_in, hidden, batch_first=True)
        self.out = nn.Linear(hidden + n_static, n_out)
    def forward(self, x, s=None):                # x: batch x time x features
        h, _ = self.rnn(x)
        z = h[:, -1] if s is None else torch.cat([h[:, -1], s], 1)
        return self.out(z)
# training step: loss.backward(); nn.utils.clip_grad_norm_(model.parameters(), 1.0); opt.step()
```

- Full code: lectures/L5/analysis_L5.py and the notebook of this lecture.