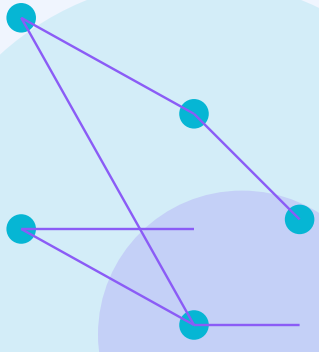
RNN TA LSTM

Рекурентні нейронні мережі

Лекція 9



LSTM

архітектура

ETH

прогноз

Python

TensorFlow

ПЛАН

ВІД MLP ДО РЕКУРЕНТНИХ МЕРЕЖ

MLP (багатошаровий перцептрон)

- Статична архітектура - без пам'яті
- Обробляє кожен вхід незалежно
- Не враховує порядок і контекст
- Входи фіксованого розміру
- Не придатний для послідовностей

RNN (рекурентна мережа)

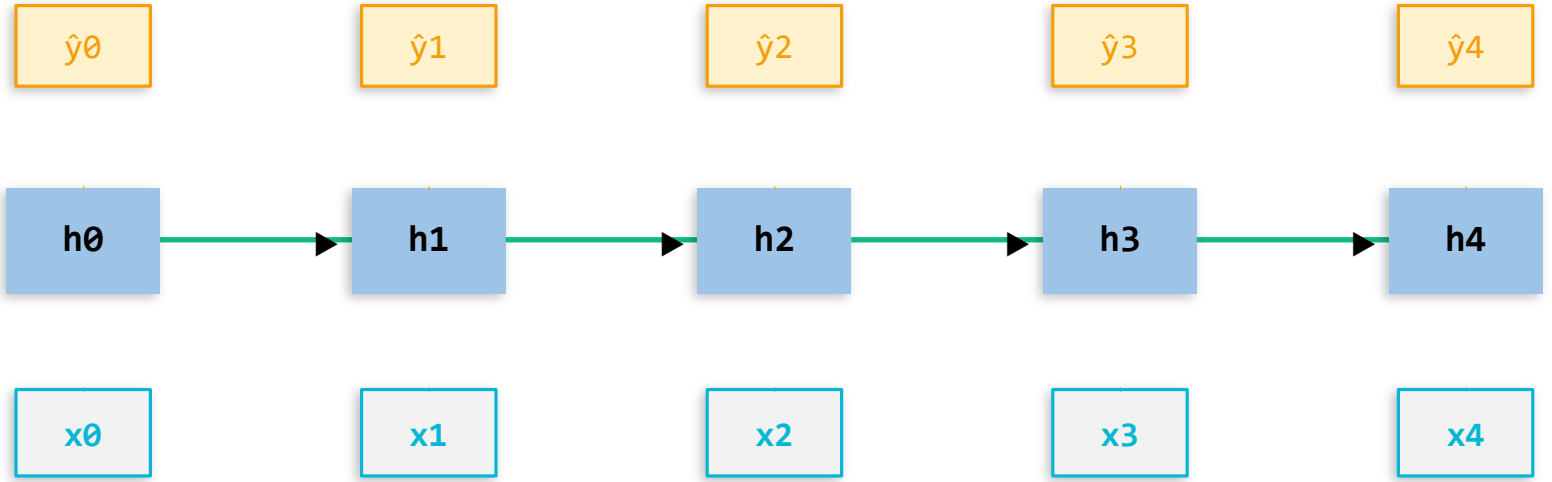
- Має прихований стан h_t - 'пам'ять'
- Обробляє послідовності x_t крок за кроком
- Ваги W спільні для всіх кроків t
- Вихід залежить від поточного та минулих входів
- Ідеальний для часових рядів, тексту, мовлення

КЛАСИЧНИЙ RNN: МАТЕМАТИЧНА МОДЕЛЬ

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

$$\hat{y}_t = \text{softmax}(W_y \cdot h_t + b_y)$$

Розгорнута RNN у часі



Проблема: vanishing gradient при T>20 — мережа забуває далекий контекст. Рішення: LSTM

BPTT ТА ПРОБЛЕМИ НАВЧАННЯ RNN

$$\text{BPTT: } \partial L/\partial W_h = \Sigma_t \Sigma_{k \le t} \left(\Pi_j \partial h_j/\partial h_{j-1} \right) \cdot \partial h_k/\partial W_h$$

1 **Vanishing Gradient**

$$\|\partial h_t/\partial h_1\| = \|\Pi \partial h_t/\partial h_{t-1}\| \to 0$$

Добуток малих чисел → градієнт зникає. Модель не навчається на далеких залежностях.

Рішення: LSTM / GRU / Transformer

2 **Exploding Gradient**

$$\|\partial h_t/\partial h_1\| \to \infty$$

Добуток великих значень → NaN. Ваги розходяться. Модель руйнується під час навчання.

Рішення: gradient clipping: $g \leftarrow g \cdot max\_norm/\|g\|$

3 **Повільне навчання**

$O(T \cdot H^2)$ — не можна паралелізувати

Залежність від попереднього кроку унеможливлює паралельне навчання на GPU/TPU.

Рішення: Truncated BPTT; Transformer

LSTM: ІДЕЯ ТА CELL STATE

Hochreiter & Schmidhuber, 1997 — вирішення проблеми vanishing gradient

LSTM додає до звичайного RNN дві «лінії»: cell state c_t (довгострокова пам'ять — «транспортна стрічка») та hidden state h_t (короткострокова пам'ять). Три гейти (вентилі) регулюють потік інформації через cell state.

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \qquad h_t = o_t \odot \tanh(c_t)$$

Cell state c_t — 'транспортна стрічка' (довгострокова пам'ять)

Hidden state h_t

Forget Gate

σ(·) → 0..1
Що забути

Input Gate

σ(·) + tanh
Що додати

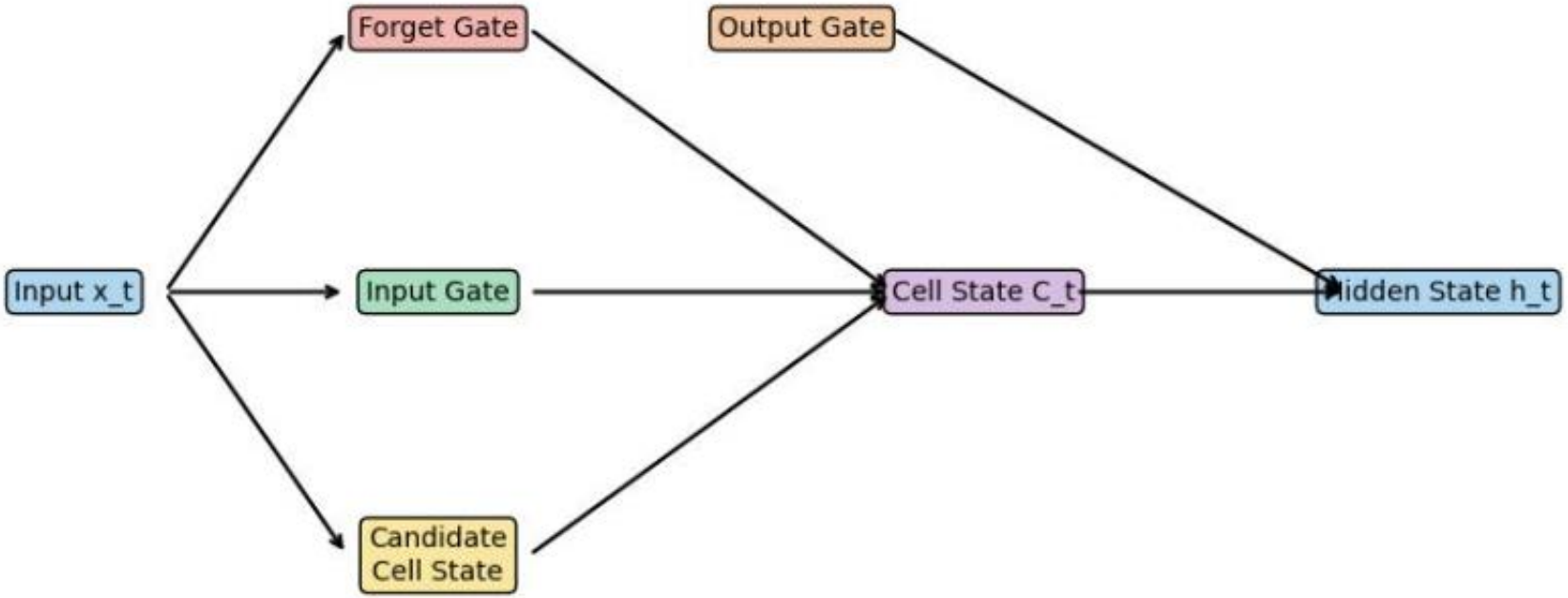
Cell Update

tanh(·)
Кандидат

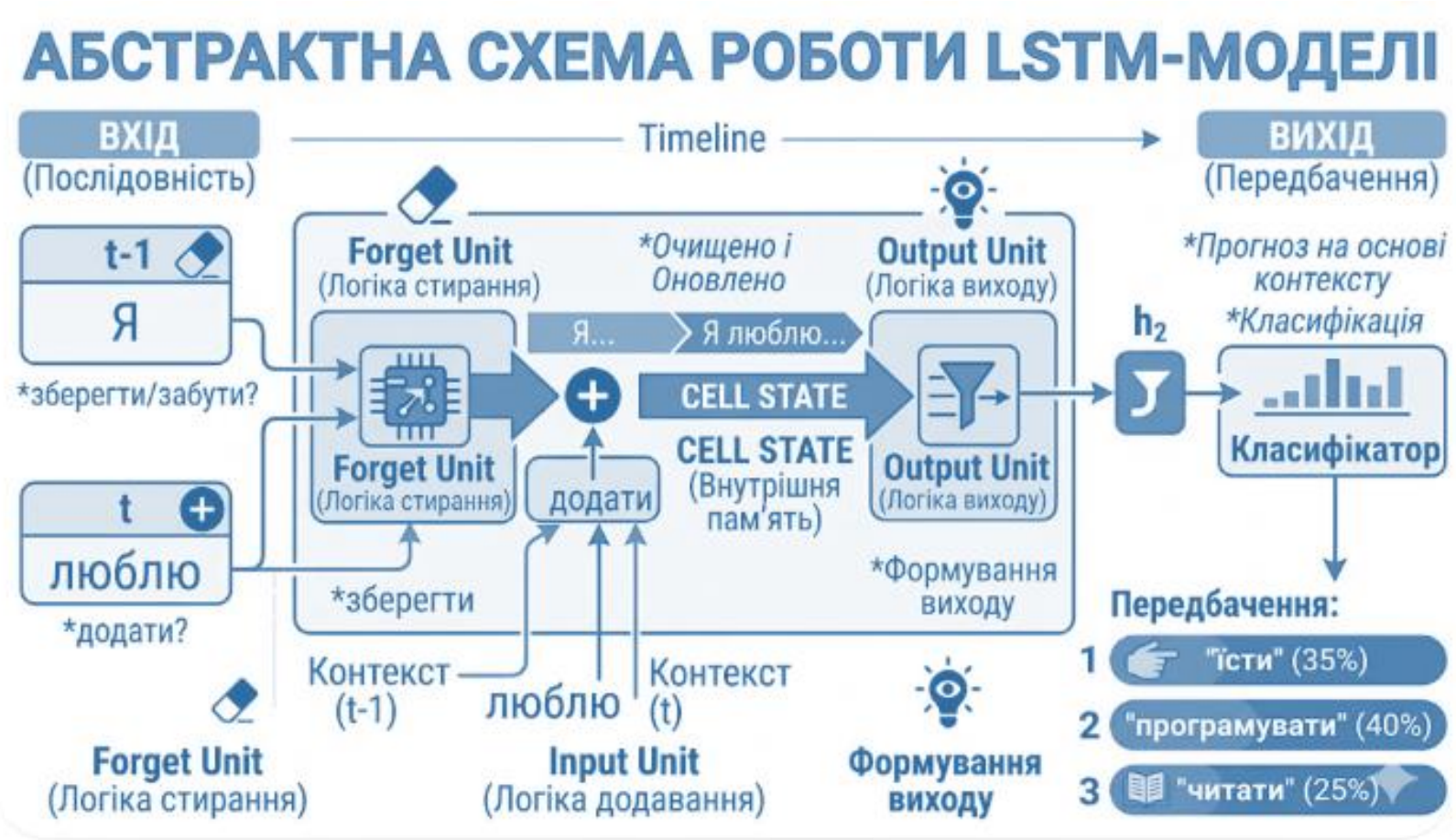
Output Gate

σ(·) + tanh
Що видати

Приклад


Forget Gate
Output Gate
Input x_t
Input Gate
Cell State C_t
idden State h_t
Candidate
Cell State

Приклад


АБСТРАКТНА СХЕМА РОБОТИ LSTM-МОДЕЛІ
ВХІД
(Послідовність)
Timeline
ВИХІД
(Передбачення)
t-1
Я
*зберегти/забути?
t
люблю
*додати?
Forget Unit
(Логіка стирання)
*Очищено і Оновлено
Output Unit
(Логіка виходу)
*Прогноз на основі контексту
*Класифікація
Я...
Я люблю...
CELL STATE
CELL STATE
(Внутрішня пам'ять)
h2
Класифікатор
Forget Unit
(Логіка стирання)
додати
Output Unit
(Логіка виходу)
*зберегти
*Формування виходу
Передбачення:
1 "їсти" (35%)
2 "програмувати" (40%)
3 "читати" (25%)
Контекст (t-1)
люблю
Контекст (t)
Forget Unit
(Логіка стирання)
Input Unit
(Логіка додавання)
Формування виходу

FORGET GATE TA INPUT GATE

Forget Gate — «що забути»

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

Sigmoid σ повертає значення 0..1 для кожної комірки cell state.

0 = повністю забути
1 = повністю зберегти

Приклад: у задачі NLP — при зустрічі нового підмета модель може скинути інформацію про попередній. $f_t \odot c_{t-1}$ масштабує стару пам'ять.

Input Gate — «що запам'ятати»

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$

Input gate i_t визначає, яку частку нових даних записати.

$\tilde{c}_t$ — кандидат у пам'ять (через tanh, діапазон −1..1).

Оновлення cell state:
$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$

Це дозволяє selectively додавати нову інформацію, не стираючи цінний давній контекст.

OUTPUT GATE ТА ПОВНА ФОРМУЛА LSTM

Output Gate — «що видати»

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \odot \tanh(c_t)$$

Output gate визначає, яку частину cell state виводити назовні як hidden state h_t.

h_t передається далі як вхід наступного кроку та на вихідний шар моделі.

Зведена таблиця гейтів LSTM

Гейт	Формула	Роль
Forget f_t	$\sigma(W_f[h,x]+b)$	Що забути
Input i_t	$\sigma(W_i[h,x]+b)$	Що записати
Cell $\tilde{c}_t$	$\tanh(W_c[h,x]+b)$	Кандидат
Output o_t	$\sigma(W_o[h,x]+b)$	Що видати
Cell c_t	$f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$	Нова пам'ять
Hidden h_t	$o_t \odot \tanh(c_t)$	Вихід

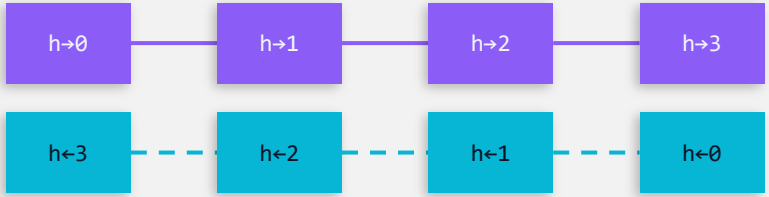
Ключовий інсайт LSTM

Cell state c_t — це «рейкова колія», яка проходить крізь всі timesteps майже без перетворень. Градієнт може текти прямо назад через цю колію, не зникаючи і не вибухаючи. Це вирішує проблему vanishing gradient класичного RNN.

ВАРІАНТИ LSTM: BIDIRECTIONAL TA STACKED

Bidirectional LSTM

Два паралельних LSTM: один читає послідовність зліва направо, другий — справа наліво. Виходи об'єднуються.



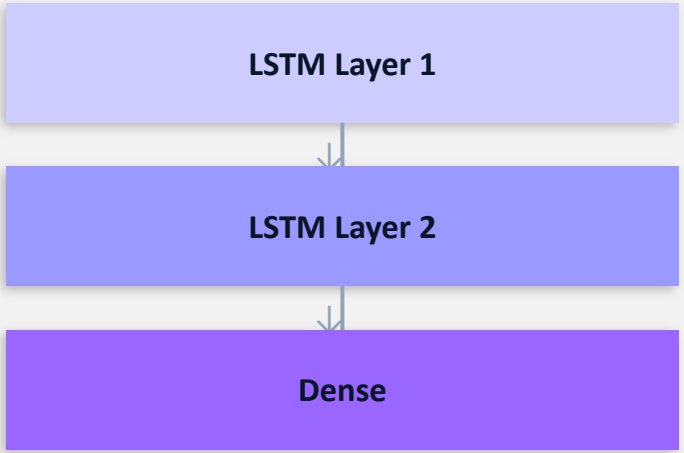
→ Forward (P2) ← Backward (A1)

Застосування: NER, аналіз тональності, machine translation (encoder).

```
h_bi = [h_forward ; h_backward]
```

Stacked LSTM (глибокий)

Кілька шарів LSTM один над одним. Вихід попереднього шару є входом наступного.



Більша глибина → складніші патерни, але потребує більше даних та часу навчання. return_sequences=True.

GRU: GATED RECURRENT UNIT (Cho, 2014)

Рівняння GRU

Update Gate

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

Reset Gate

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

Candidate

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t])$$

Output

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t$$

LSTM vs GRU

	LSTM	GRU
Гейти	3 (f,i,o)	2 (z,r)
Стани	h_t + c_t	тільки h_t
Параметри	~4×H²	~3×H²
Пам'ять	Довша	Коротша
Швидкість	Повільніше	Швидше
Рік засн.	1997	2014

ЗАСТОСУВАННЯ RNN / LSTM / GRU

Фінансові часові ряди

Прогноз цін акцій, криптовалют, валютних курсів. Many-to-One або Many-to-Many.

Машинний переклад

Seq2Seq: LSTM-encoder кодує речення, decoder генерує переклад. Google Translate до 2017.

Розпізнавання мовлення

CTC + LSTM → аудіо до тексту. DeepSpeech, Whisper (частково).

Медичні сигнали

ЕКГ, ЕЕГ — виявлення аномалій. ICU прогноз стану пацієнта.

Метеорологія

Прогноз погоди, температури, опадів. Many-to-Many з часовим вікном.

NLP задачі

Аналіз тональності, NER, text classification. Bidirectional LSTM.

ПРАКТИЧНА РОБОТА №6: ПОСТАНОВКА ЗАДАЧІ

Тема: Регресія. Прогноз ціни Ethereum (ETH)

Мета:
Набути практичних навичок побудови LSTM-моделі для прогнозування цін на валютному ринку блокчейну

Стек технологій:

TensorFlow 2.x

побудова та навчання LSTM-моделі

Keras Sequential

LSTM, Dense, Dropout шари

Google Colab

хмарне середовище з GPU

Pandas / NumPy

завантаження та обробка даних

Matplotlib

візуалізація ціни та прогнозу

Scikit-learn

метрики MAE та MSE

ДАТАСЕТИ ТА ДЖЕРЕЛА ДАНИХ

historical_data.csv

Джерело: Binance

Поля:

- close — ціна закриття (цільова змінна)
- open, high, low — OHLC дані
- volume — обсяг торгів
- close_time — мітка часу

Основний датасет. Містить цінові та об'ємні дані.

blockchain_data.csv

Джерело: Blockchain.com

Поля:

- tx_count — кількість транзакцій
- hash_rate — потужність мережі
- difficulty — складність майнінгу
- time — мітка часу

Метрики мережі Ethereum — on-chain дані.

twitter_sentiments.csv

Джерело: Twitter API

Поля:

- positive — частка позитивних твітів
- negative — частка негативних
- neutral — нейтральні згадки
- time — мітка часу

Сентиментальний аналіз твітів про ETH.

```
merged_df = pd.merge(pd.merge(blockchain_df, prices_df, left_on='time',
right_on='close_time'), twitter_sentiments, on='time')  → inner merge
```

СХЕМА ВИКОНАННЯ ПРАКТИЧНОЇ РОБОТИ

1 **Завантаження даних**

historical_data.csv,
blockchain_data.csv,
twitter_sentiments.csv

2 **Об'єднання датасетів**

pd.merge → inner merge → лише дати з усіх 3 джерел

3 **Очищення та сортування**

Видалення NaN, сортування за часом, вибір ознак

4 **Нормалізація (zero-base)**

$x\_norm = x/x_0 - 1$ → відносні зміни у вікні

5 **Sliding Window (10 кроків)**

extract_window_data → X_train, X_test масиви

6 **Побудова LSTM моделі**

LSTM(256) → Dropout(0.2) → Dense(1) → Activation('linear')

7 **Callbacks: EarlyStopping + LRScheduler**

patience=10, lr×0.75 кожні 10 епох

8 **Навчання на 4 комбінаціях ознак**

[hist], [hist+chain], [hist+twitter], [all 3]

9 **Оцінка: MSE та візуалізація**

plot_result() → порівняння actual vs predicted

НОРМАЛІЗАЦІЯ ДАНИХ: ZERO-BASE

$x\_norm = x / x_0 - 1$ де x_0 — перший елемент вікна

Чому zero-base краще ніж MinMax?

MinMax (x-min)/(max-min) нормалізує відносно глобального мінімуму/максимуму — вони змінюються з новими даними.
Zero-base нормалізує ВСЕРЕДИНІ вікна — кожне вікно стає незалежним відносним відхиленням від початкової точки. Масштаб не залежить від абсолютної ціни.

Метод ковзного вікна (Sliding Window)

window_len = 10 кроків.

Кожне вікно: $[t_0, t_1, \ldots, t_9] \rightarrow$ прогноз y = close$[t_{10}]$

Одне вікно після іншого з кроком 1. Результат: масив X_train shape=(N, 10, features)

```
def normalise_zero_base(df):
    coefs = df.iloc[0].values
    for i in range(len(coefs)):
        if coefs[i] == 0: coefs[i] = 1
    return df / coefs - 1
```

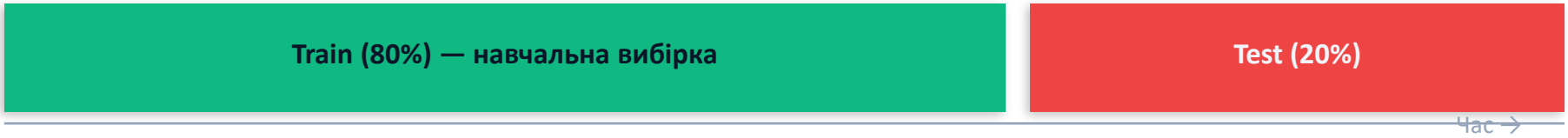
```
def extract_window_data(df, window_len):
    window_data = []
    for idx in range(len(df) - window_len):
        data = normalise_zero_base(df[idx:
(idx + window_len)].copy())
        window_data.append(data.values)
    return np.array(window_data)  # shape:
(N, window_len, features)
```

РОЗБИТТЯ ДАНИХ: TRAIN / TEST SPLIT

ВАЖЛИВО: часові ряди НЕ можна перемішувати перед розбиттям!

sklearn.train_test_split(shuffle=True) порушує хронологічний порядок → модель 'бачить' майбутнє під час навчання → data leakage

```
def train_test_split(df, test_size=0.2):
    split_row = len(df) - int(test_size * len(df))
    train_data = df.iloc[:split_row]   # перші 80% — навчальні
    test_data  = df.iloc[split_row:]   # останні 20% — тестові
    return train_data, test_data       # хронологічний порядок збережено!
```



prepare_data() — загальний пайплайн підготовки:

```
train_test_split → extract_window_data(X_train, X_test) → normalize target y_train = y/y_prev - 1
```

CALLBACKS: EARLYSTOPPING TA LR SCHEDULER

EarlyStopping

```
early_stopping_callback =
  tf.keras.callbacks.EarlyStopping(
    patience=10,
    restore_best_weights=True
  )
```

patience=10: якщо val_loss не покращується 10 епох поспіль — зупиняємось.

restore_best_weights=True: повертаємо ваги найкращої епохи, а не останньої.

Це запобігає перенавчанню (overfitting) та заощаджує час навчання.

LearningRateScheduler

```
def lr_scheduler(epoch, lr):
    if epoch > 0 and epoch % 10 == 0:
        return lr * 0.75
    return lr

lr_scheduler_callback =
  tf.keras.callbacks.LearningRateScheduler(
    lr_scheduler)
```

Кожні 10 епох learning rate зменшується на 25%.

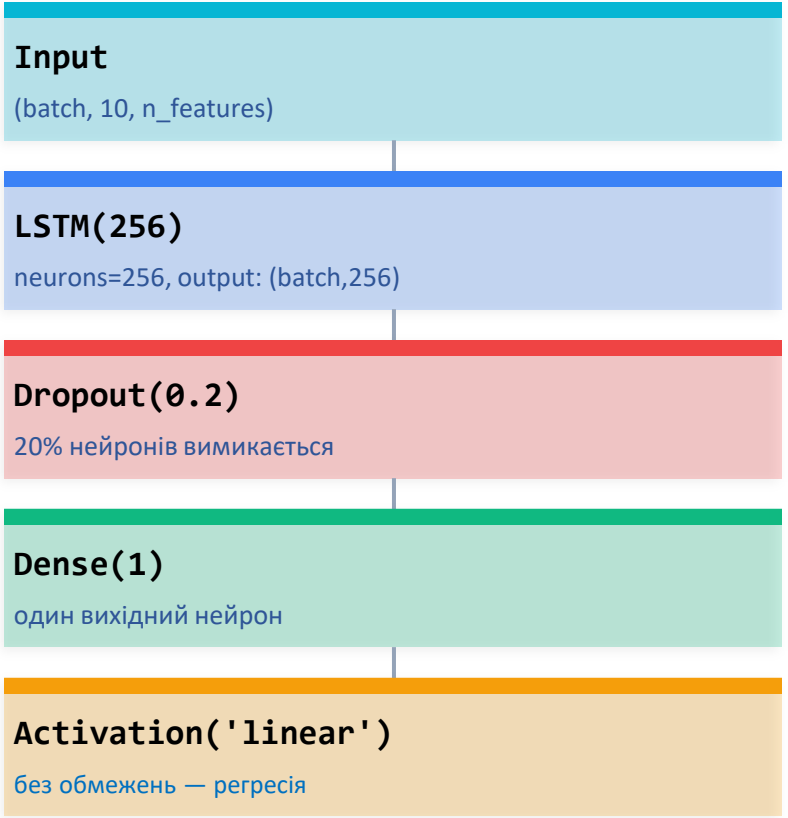
Початок: великий lr → швидке наближення до мінімуму.
Середина: середній lr → точне наближення.
Кінець: малий lr → точне влучення в мінімум.

Це запобігає 'перестрибуванню' мінімуму функції втрат.

LSTM МОДЕЛЬ: КОД ТА АРХІТЕКТУРА

```
def build_lstm_model(input_data, neurons,
                     dropout, optimizer):
    model = Sequential()
    model.add(LSTM(neurons,
        input_shape=(input_data.shape[1],
                     input_data.shape[2])))
    model.add(Dropout(dropout))
    model.add(Dense(units=1))
    model.add(Activation('linear'))
    model.compile(loss='mse',
                  optimizer=optimizer)
    return model
```

```
window_len = 10   # кількість кроків у вікні
neurons    = 256  # нейрони LSTM шару
dropout    = 0.2  # 20% dropout
optimizer  = 'adam'
epochs     = 100  # max epochs
batch_size = 32   # батч
```



НАВЧАННЯ НА 4 КОМБІНАЦІЯХ ОЗНАК

```
data_sets = [['historical'], ['historical','blockchain'],
             ['historical','twitter'], ['historical','blockchain','twitter']]

for item in data_sets:
    fields = []
    for field_type in item:
        fields.extend(data_props[field_type])      # збираємо ознаки
    df = merged_df[fields]
    train, test, X_train, X_test, y_train, y_test = prepare_data(df, window_len)
    current_model = build_lstm_model(X_train, neurons, dropout, optimizer)
    current_model.fit(np.asarray(X_train).astype('float32'),
                      np.asarray(y_train).astype('float32'),
                      epochs=epochs, batch_size=batch_size, shuffle=True,
                      callbacks=[early_stopping_callback, lr_scheduler_callback])
```

Комбінація 1 1

historical

close, open, high, low, volume

Комбінація 2 2

historical + blockchain

+ tx_count, hash_rate, difficulty

Комбінація 3 3

historical + twitter

+ positive, negative, neutral

Комбінація 4 4

historical + blockchain + twitter

Всі ознаки разом (найбільш повна)

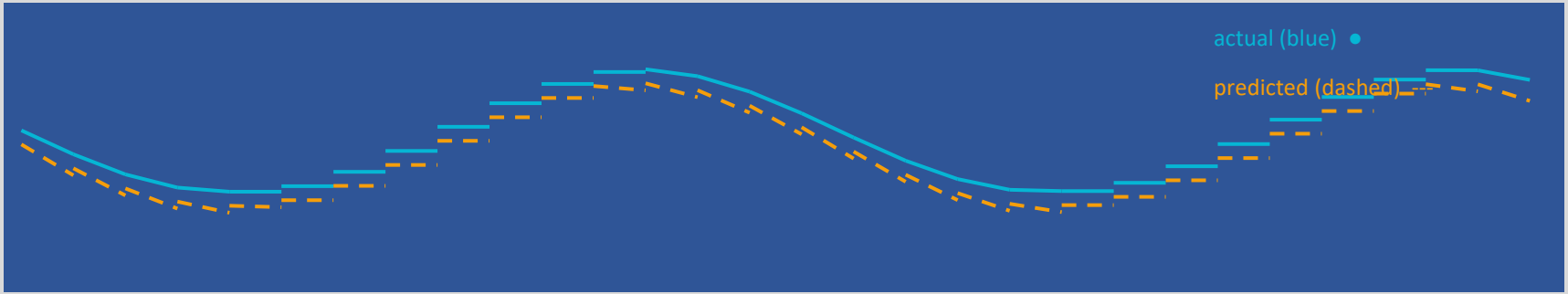
`mse_results[key] = mean_squared_error(preds, y_test)` → рейтинг комбінацій за MSE

ФУНКЦІЇ ВІЗУАЛІЗАЦІЇ РЕЗУЛЬТАТІВ

```
def line_plot(train, test, label1=None, label2=None,
              title='', lw=2):
    fig, ax = plt.subplots(1, figsize=(13,7))
    if 'time' in train.keys():
        ax.plot(train['time'].apply(
          lambda x: datetime.fromtimestamp(x)),
          train['close'], label=label1, lw=lw)
    else:
        ax.plot(train['close'],
                label=label1, lw=lw)
    ax.set_ylabel('Ціна [USD]', fontsize=14)
```

```
def plot_result(actual, predicted):
    fig, ax = plt.subplots(1,figsize=(13,7))
    ax.plot(actual, label='actual', lw=1)
    for key in predicted.keys():
        ax.plot(predicted[key],
                label=key, lw=1)
    ax.set_ylabel('Ціна [USD]', fontsize=14)
    ax.legend(loc='best', fontsize=16)
    plt.show()
```

Приклад виводу: графік actual vs predicted



МЕТРИКИ ОЦІНКИ ЯКОСТІ РЕГРЕСІЇ

MSE

Mean Squared Error

$$\frac{1}{n} \cdot \Sigma (y_i - \hat{y}_i)^2$$

Чутлива до великих відхилень. Штрафує викиди сильніше. Використовується як loss function.

RMSE

Root Mean Squared Error

$$\sqrt{MSE}$$

В одиницях цільової змінної (USD). Легше інтерпретувати ніж MSE.

MAE

Mean Absolute Error

$$\frac{1}{n} \cdot \Sigma |y_i - \hat{y}_i|$$

Стійка до викидів. Не квадратує помилки. Всі відхилення рівнозначні.

R^2

Коефіцієнт детермінації

$$1 - SS_{res} / SS_{tot}$$

Частка поясненої дисперсії. 1.0 = ідеально. 0 = модель нічого не пояснює.

```
rating = sorted(mse_results.items(), key=lambda x: x[1])
for item in rating: print(item[0], '-', item[1])
```

РЕЗУЛЬТАТИ ТА ІНТЕРПРЕТАЦІЯ

Комбінація ознак	MSE (типово)	Очікуваний результат
historical (тільки ціна)	~0.0025–0.004	Базова лінія
historical + blockchain	~0.002–0.003	Покращення on-chain даних
historical + twitter	~0.0018–0.003	Сентимент дає додатковий сигнал
historical + blockchain + twitter	~0.0015–0.0025	Найкраще (всі сигнали)

Аналіз plot_result(): На що звертати увагу: (1) Чи слідує прогноз за трендом? (2) Як великі відхилення в точках різких злетів/падінь? (3) Чи є систематичне запізнення прогнозу?

Висновки по ознаках: Рейтинг ознак: twitter sentiment зазвичай дає кращий буст, ніж blockchain дані. Але все залежить від ринкового режиму (bull/bear market).

Overfitting ознаки: Якщо train_loss << val_loss — є перенавчання. Збільшити dropout, зменшити neurons, або додати більше даних. EarlyStopping має це зловити.

ЗАВДАННЯ 2 ТА 3: РОЗШИРЕННЯ

Завдання 2. Мультикрокове прогнозування

Замість 1 кроку — прогноз на 7 і 30 днів вперед.

Підхід Direct Multi-step:
Навчити окрему модель для кожного горизонту h.
$y_{t+h} = f(X_t, ..., X_{t-w})$

Підхід Recursive:
Використати прогноз $\hat{y}_{t+1}$ як вхід для $\hat{y}_{t+2}$ і т.д.

Очікуване: MSE збільшується зі зростанням горизонту. MAE також зростає, але повільніше завдяки стійкості до викидів.

Завдання 3. Аналіз та інтерпретація

Аналіз важливості ознак

Прибирати по одній групі (ablation study) і фіксувати зміну MSE. Побудувати рейтинг впливу.

Аналіз помилок у часі

Графік $|y - \hat{y}|$ по датах. Ідентифікувати ринкові події (crash/pump) де помилка максимальна.

Monte Carlo Dropout

100 forward passes із увімкненим dropout. Будуємо довірчий інтервал [5th, 95th percentile] — невизначеність прогнозу.

ЗАВДАННЯ 4 ТА ДОДАТКОВІ ЗАВДАННЯ

Завдання 4

Побудувати модель прогнозу Bitcoin (BTC). Датасет вибрати самостійно з https://www.kaggle.com

Додаткові завдання (для оцінки «відмінно»):

A — Порівняння архітектур

Побудувати LSTM, GRU та Bidirectional LSTM. Порівняти MSE і час навчання.

B — Стековий LSTM

2 шари LSTM один над одним (return_sequences=True для першого). Дослідити вплив глибини.

C — Технічні індикатори

Додати RSI, MACD, Bollinger Bands як ознаки. Перевірити покращення MSE.

D — Крос-валютні ознаки

Додати ціну BTC як ознаку при прогнозуванні ETH. Перевірити кореляцію активів.

E — Keras Tuner / Optuna

Автоматичний підбір neurons, dropout, window_len, batch_size. Порівняти з базовою моделлю.

F — Transformer замість LSTM

Реалізувати self-attention для часових рядів. Порівняти з LSTM за точністю та часом.

ВИСНОВКИ ТА КОНТРОЛЬНІ ЗАПИТАННЯ

RNN та LSTM

- RNN: $h_t=\tanh(W_h h_{t-1}+W_x x_t+b)$
- Vanishing gradient при T>20
- LSTM: 3 гейти + cell state c_t
- $c_t=f_t\odot c_{t-1}+i_t\odot \tilde{c}_t$, $h_t=o_t\odot\tanh(c_t)$

Практична робота №6

- 3 датасети: hist+chain+twitter
- Zero-base нормалізація
- Sliding window (10 кроків)
- 4 комбінації ознак → рейтинг MSE

Метрики та Callbacks

- MSE = $(1/n)\Sigma(y-\hat{y})^2$ — loss fn
- MAE = $(1/n)\Sigma|y-\hat{y}|$ — стійка
- EarlyStopping (patience=10)
- LRScheduler (×0.75 кожні 10 епох)