Recurrent Neural Networks and the Exploding and Vanishing Gradient Problems

Nhóm 6

Ngày 8 tháng 1 năm 2026

Thành viên nhóm

Lê Quang Trung — MSSV: 24521883

Nguyễn Ngọc Hưng — MSSV: 24520610

Trần Lê Anh Pha — MSSV:

Mục lục chương 1

Sequential Data

Dữ liệu chuỗi (sequential data) là các phần tử được sắp theo thứ tự tạo thành chuỗi.

Ví dụ: chuỗi thời gian, chuỗi DNA, hoặc chuỗi hành vi người dùng.

Mô hình chuỗi (sequence models) nhận/ xuất dữ liệu dạng chuỗi.

What Are Recurrent Neural Networks?

Recurrent Neural Networks (RNNs) là kiến trúc mạng nơ-ron dùng để phát hiện quy luật trong dữ liệu chuỗi.

- Có thể áp dụng cho ảnh sau khi tách ảnh thành các patch có thứ tự.
- Điểm khác biệt với feedforward neural networks (MLP) là cách thông tin được truyền qua thời gian.

Model Architecture

RNN có chu trình (cycles) truyền thông tin ngược trong mạng, cho phép mô hình xét các input trước đó $X_{0:t-1}$ thay vì chỉ input hiện tại X_t.

Model Architecture

Gọi hidden state tại thời điểm t là H_t và input là X_t. Các tham số được chia sẻ: W_{xh}, W_{hh}, W_{ho}, cùng bias b_h, b_o. Activation function ϕ thường là logistic sigmoid hoặc tanh.

$$H_t = \phi_h(X_t W_{xh} + H_{t-1} W_{hh} + b_h)\,,$$
$$O_t = \phi_o(H_t W_{ho} + b_o)\,.$$

Loss Function

Hàm loss đánh giá hiệu năng bằng cách so sánh output y_t với target z_t:

$$\mathcal{L}(y, z) = \sum_{t=1}^{T} \ell_t(y_t, z_t).$$

Loss phụ thuộc bài toán; ví dụ: Euclidean distance, Hamming distance, cross-entropy.

Pros and Cons

Ưu điểm

Xử lý chuỗi độ dài bất kỳ, chia sẻ weights qua thời gian.

Kích thước mô hình không tăng theo độ dài chuỗi.

Tính toán xét đến thông tin quá khứ.

Nhược điểm

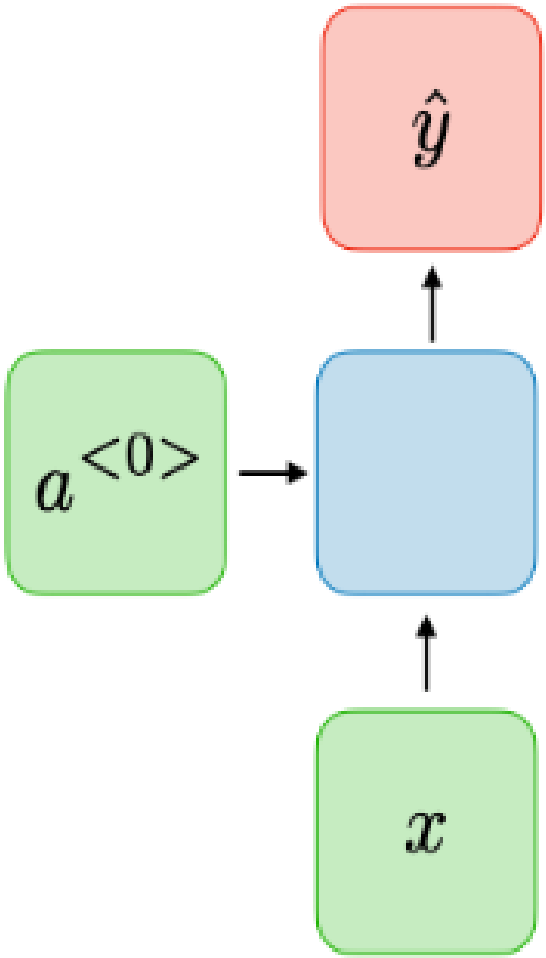
Tính toán có thể chậm.

Khó lấy thông tin ở quá khứ xa.

Không xét được input tương lai khi tạo state hiện tại.

Various Usage of RNNs: One-to-One

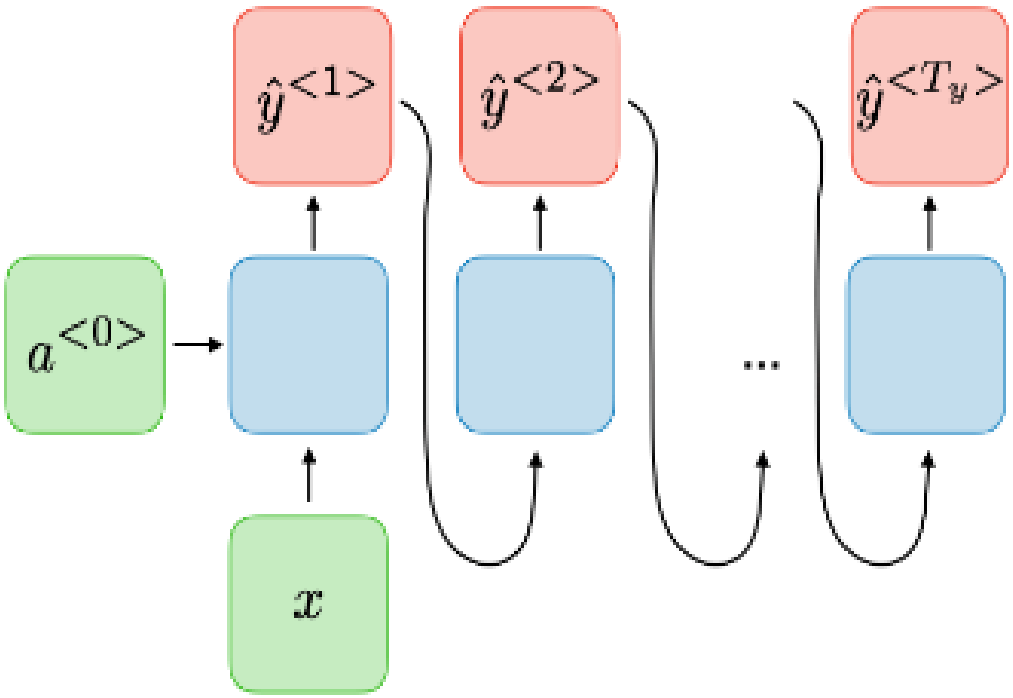
One-to-One RNNs (vanilla neural networks) dùng cho tác vụ một input, một output.



Various Usage of RNNs: One-to-Many

One-to-Many RNNs sinh nhiều output từ một input.

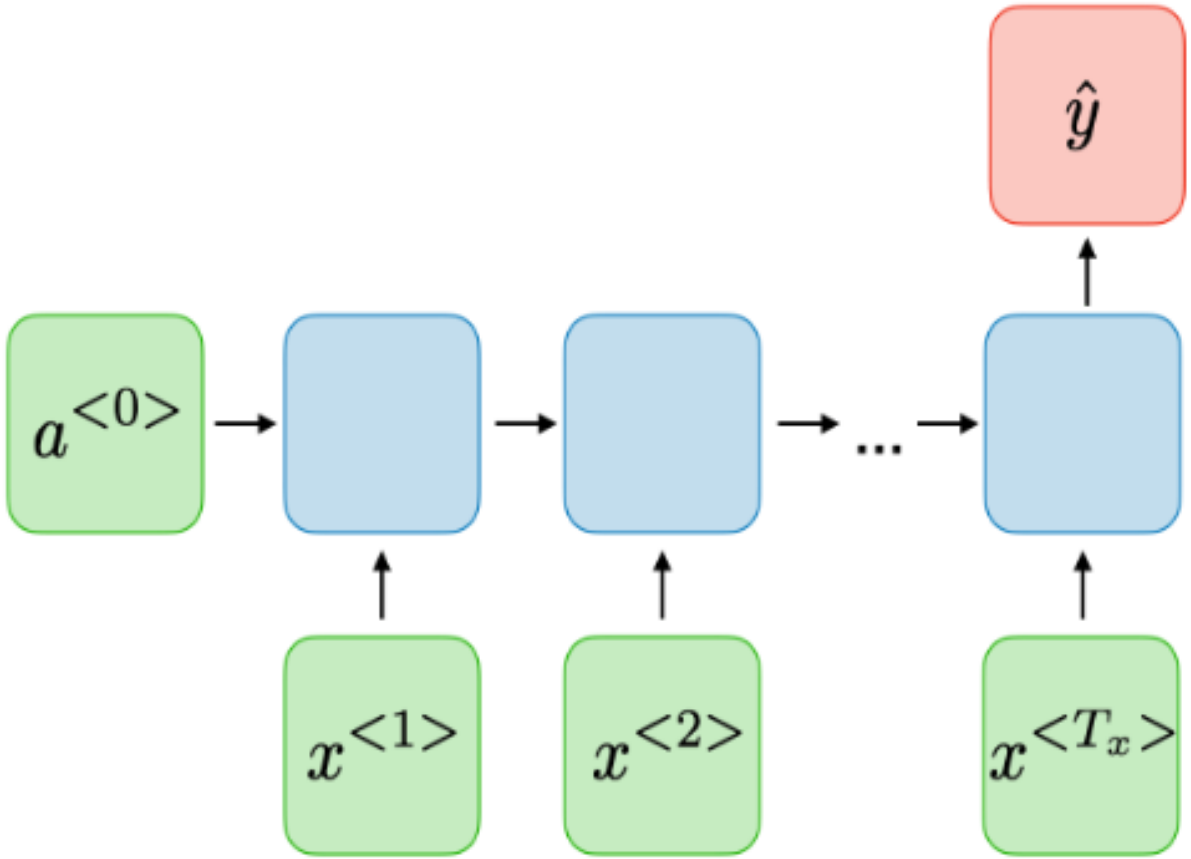
Ứng dụng: music generation, image captioning.



Various Usage of RNNs: Many-to-One

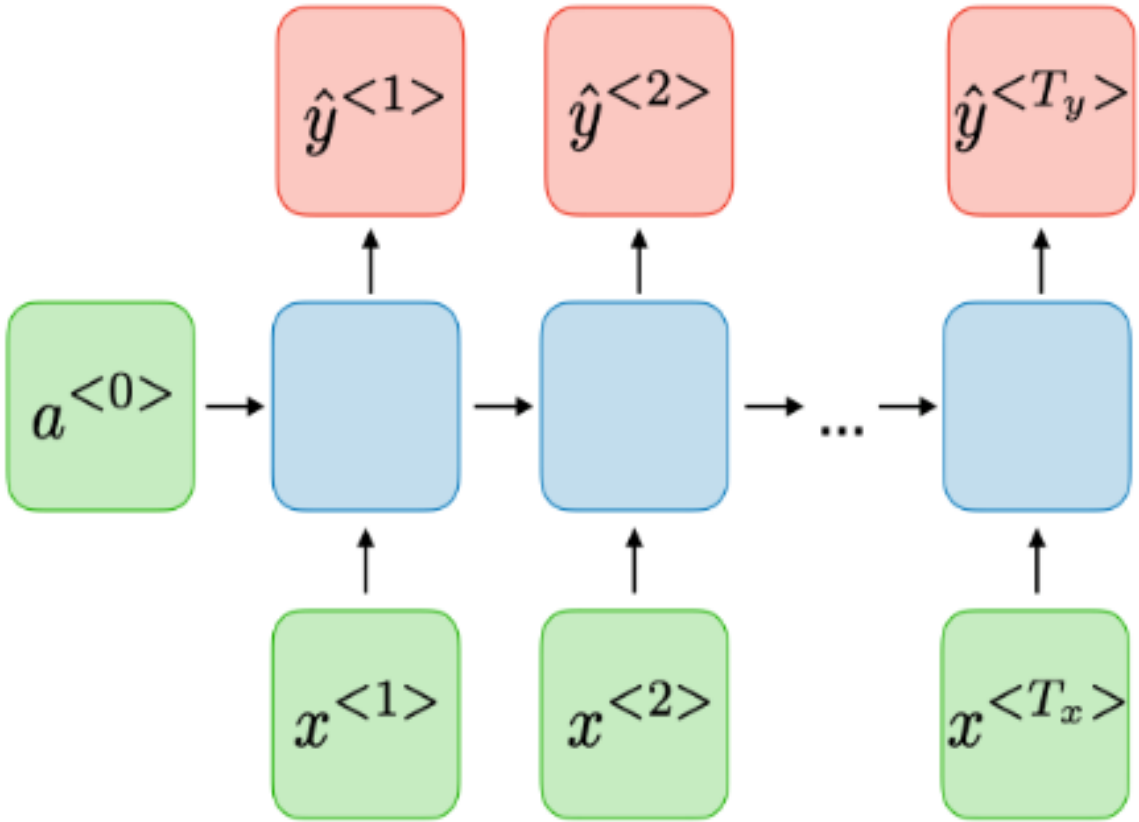
Many-to-One RNNs nhận chuỗi input và xuất một output cố định.

Ví dụ phổ biến: sentiment analysis.



Various Usage of RNNs: Many-to-Many

Many-to-Many RNNs ánh xạ chuỗi input sang chuỗi output.
Số bước bằng nhau: số input = số output (ví dụ: named-entity recognition).
Số bước khác nhau: độ dài input khác output (ví dụ: machine translation).



Mục lục chương 2

Yêu cầu bổ sung: Các hình vẽ và kết quả demo cần được nhúng vào trong slides.

Thiết lập ký hiệu & công thức sẽ dùng

Input/hidden/output: $\boldsymbol{x}_t$ (đầu vào), $\boldsymbol{h}_t$ (hidden state), $\boldsymbol{o}_t$ (đầu ra).

Tiền kích hoạt hidden: $\boldsymbol{a}_t = \boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h$.

Hidden: $\boldsymbol{h}_t = \phi_h(\boldsymbol{a}_t)$, đạo hàm ϕ_h' (ví dụ tanh).

Tiền kích hoạt output: $\boldsymbol{z}_t = \boldsymbol{W}_{qh}\boldsymbol{h}_t + \boldsymbol{b}_q$.

Output: $\boldsymbol{o}_t = \phi_o(\boldsymbol{z}_t)$, với ϕ_o là activation/output layer (tuyến tính, softmax, sigmoid... tùy bài toán).

Loss chuỗi: $\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)$.

Quy trình huấn luyện RNN

Quy trình thường sẽ như thế này:

- Forward Pass: tính toán đầu ra và sai số (Loss).
- Backward Pass: tính gradient ∇J.
- Update Rule (Optimizer): dùng gradient để cập nhật tham số với các lựa chọn phổ biến:
  - Batch Gradient Descent
  - Stochastic Gradient Descent
  - Mini-batch Gradient Descent

Forward pass: tính toán qua thời gian

Khởi tạo $\boldsymbol{h}_0 = 0$.

Với mỗi bước $t = 1, \ldots, T$:

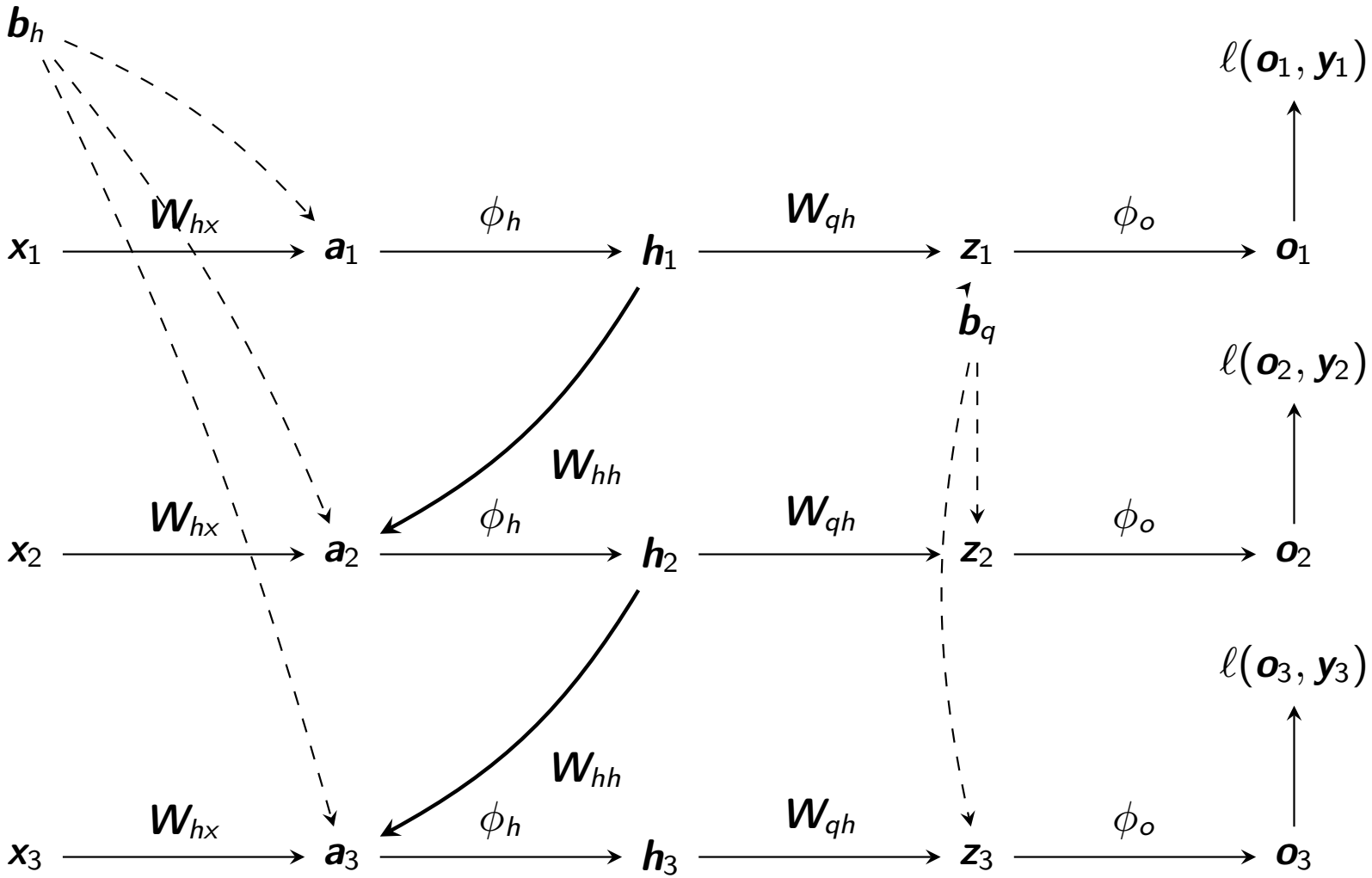
$$\boldsymbol{a}_t = \boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h, \qquad \boldsymbol{h}_t = \phi_h(\boldsymbol{a}_t)$$

$$\boldsymbol{z}_t = \boldsymbol{W}_{qh}\boldsymbol{h}_t + \boldsymbol{b}_q, \qquad \boldsymbol{o}_t = \phi_o(\boldsymbol{z}_t)$$

Tính loss tổng/ trung bình:

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)$$

Computational graph



Computational graph

Hai insight quan trọng khi lấy đạo hàm theo tham số:

Weight Sharing: cùng một tham số W_{hx}, W_{hh}, W_{qh} xuất hiện ở mọi timestep, nên

$$\frac{\partial \mathcal{L}}{\partial W} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}_t}{\partial W}.$$

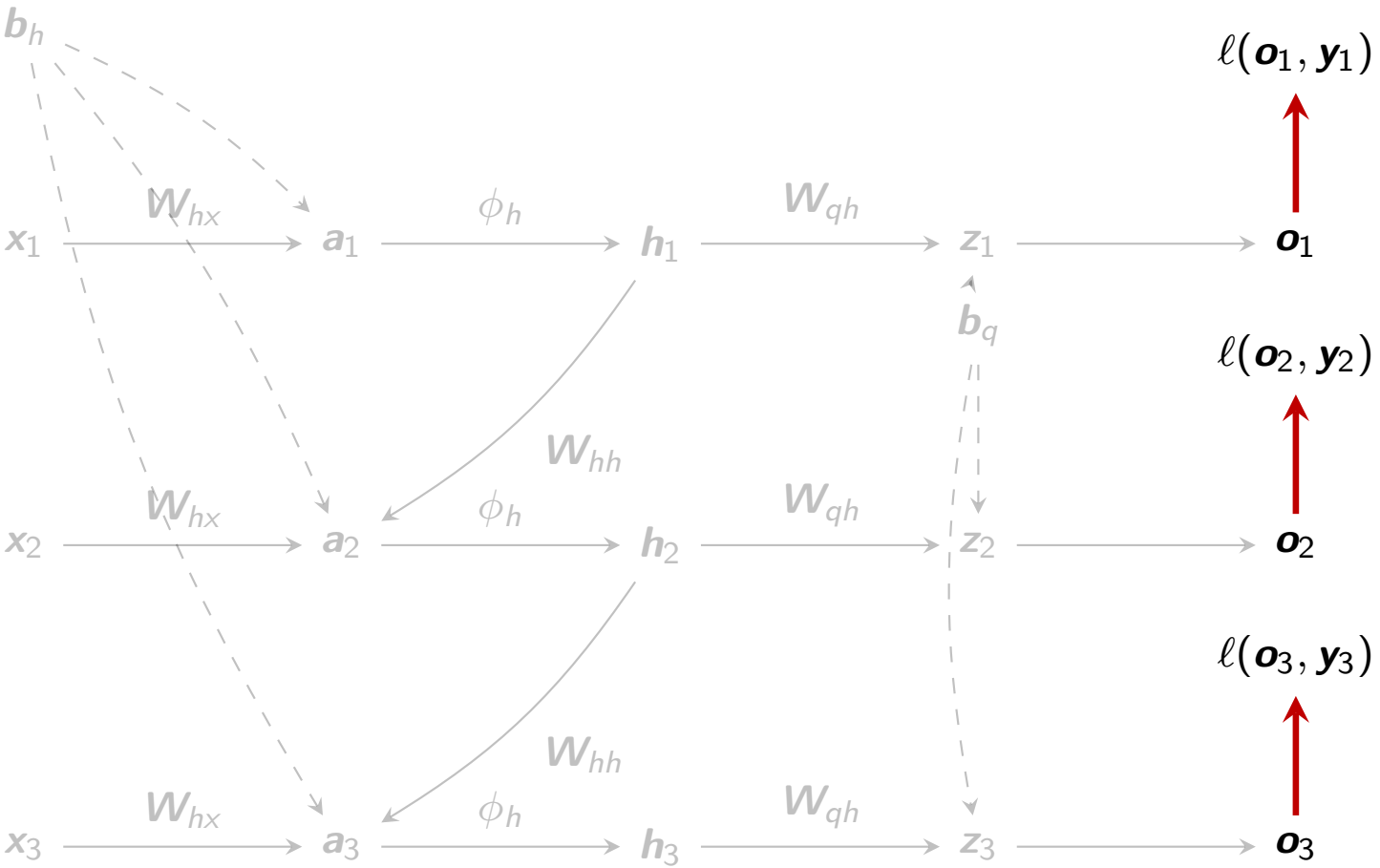
Phụ thuộc qua thời gian: gradient theo hidden tại thời điểm t nhận đóng góp từ loss hiện tại và từ tương lai:

$$\frac{\partial \mathcal{L}}{\partial \mathrm{h}_t} = \frac{\partial \mathcal{L}_t}{\partial \mathrm{h}_t} + W_{hh}^{\top} \frac{\partial \mathcal{L}}{\partial \mathrm{a}_{t+1}}.$$

Đường đạo hàm cho o_t

Mỗi o_t chỉ nối với loss tại cùng thời điểm: $o_t \to \ell_t$.

Không có cộng dồn qua các bước thời gian (chỉ nhân hệ số $\frac{1}{T}$).



Gradient theo output đầu ra o_t

Loss chuỗi: $\mathcal{L} = \frac{1}{T} \sum_{k=1}^{T} \ell(\boldsymbol{o}_k, \boldsymbol{y}_k)$.

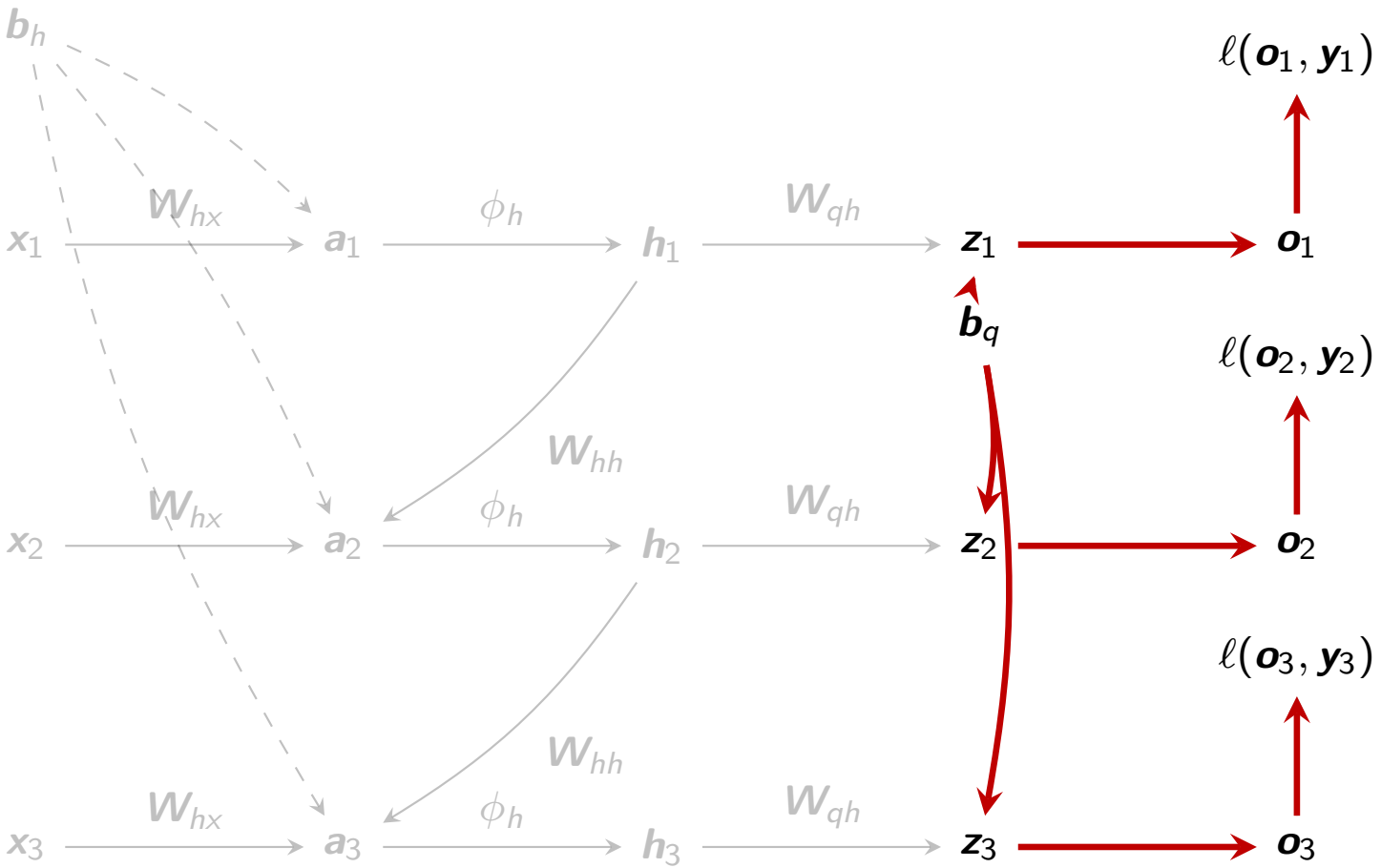
Mỗi thời điểm xuất hiện đúng một lần trong tổng nên:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t} = \frac{1}{T} \frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \in \mathbb{R}^q.$$

Đường đạo hàm cho b_q

Gradient theo $\boldsymbol{b}_q$ đi dọc nhánh $\boldsymbol{b}_q \to \boldsymbol{z}_t \to \boldsymbol{o}_t \to \ell_t$ tại từng bước thời gian.

Vì $\boldsymbol{b}_q$ được chia sẻ, các đóng góp này được cộng lại rồi chia T.



Gradient theo bias đầu ra b_q

$$\boldsymbol{o}_t = \phi_o(\boldsymbol{z}_t), \quad \boldsymbol{z}_t = \boldsymbol{W}_{qh}\boldsymbol{h}_t + \boldsymbol{b}_q,$$
$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T} \ell(\boldsymbol{o}_t, \boldsymbol{y}_t).$$

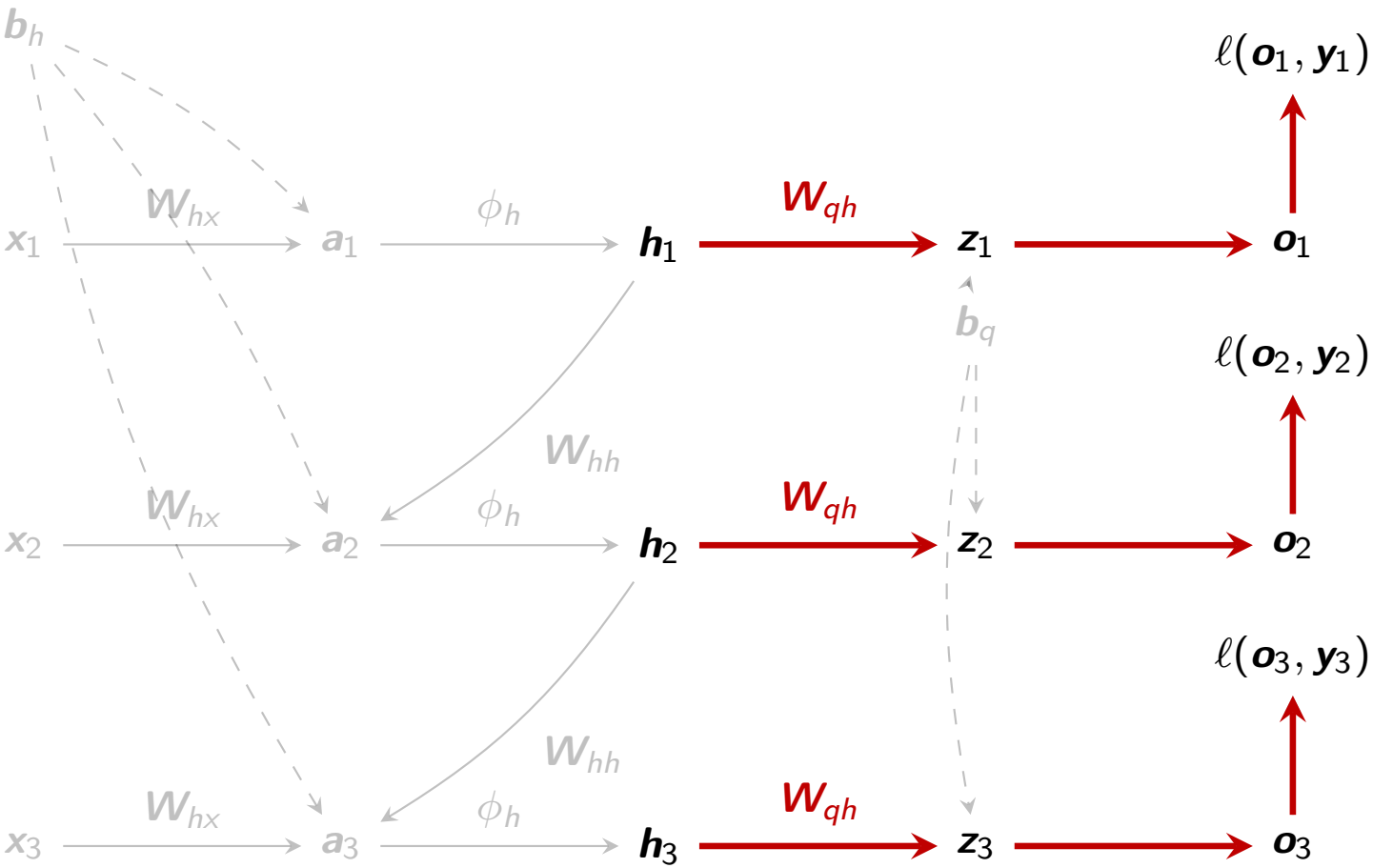
Vì $\boldsymbol{b}_q$ dùng chung cho mọi bước thời gian:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_q} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \cdot \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{z}_t} \cdot \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{b}_q} \quad \text{(chuỗi phụ thuộc } \boldsymbol{b}_q \to \boldsymbol{z}_t \to \boldsymbol{o}_t \to \ell\text{)}$$
$$= \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \odot \phi_o'(\boldsymbol{z}_t) \qquad \text{(do } \partial \boldsymbol{z}_t / \partial \boldsymbol{b}_q = I\text{)}$$

Đường đạo hàm cho W_{qh}

Gradient đi theo nhánh $W_{qh} \to z_t \to o_t \to \ell_t$, được nhân thêm $h_t^\top$ tại mỗi bước.

Vì W_{qh} dùng chung, cộng các đóng góp qua thời gian rồi chia T.



Gradient theo trọng số W_{qh}

Sử dụng $z_t = W_{qh} h_t + b_q$ và áp dụng chain rule từng bước:

$$\frac{\partial \mathcal{L}}{\partial W_{qh}} = \frac{1}{T} \sum_{t=1}^{T} \frac{\partial \ell(o_t, y_t)}{\partial W_{qh}}$$

(tách tổng theo từng thời điểm)

$$= \frac{1}{T} \sum_{t=1}^{T} \frac{\partial \ell(o_t, y_t)}{\partial o_t} \cdot \frac{\partial o_t}{\partial z_t} \cdot \frac{\partial z_t}{\partial W_{qh}}$$

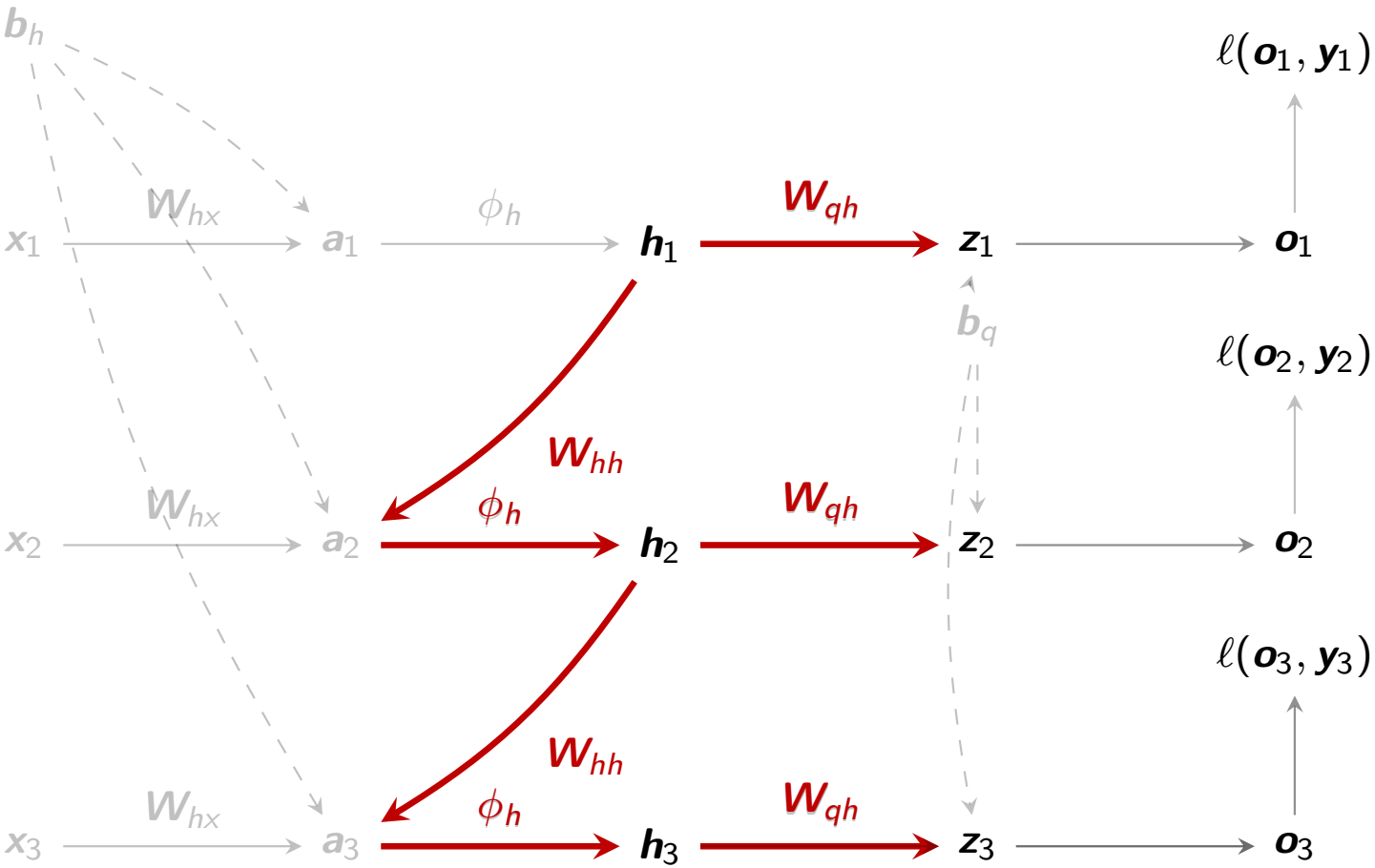
(chuỗi phụ thuộc $W_{qh} \to z_t \to o_t \to \ell$

$$= \frac{1}{T} \sum_{t=1}^{T} \left(\frac{\partial \ell(o_t, y_t)}{\partial o_t} \odot \phi_o'(z_t) \right) h_t^\top$$

Đường đạo hàm cho h_t

Hai nhánh: (1) qua output hiện tại $\boldsymbol{h}_t \to \boldsymbol{z}_t \to \boldsymbol{o}_t \to \ell_t$; (2) hồi quy về tương lai $\boldsymbol{h}_t \to \boldsymbol{a}_{t+1} \to \boldsymbol{h}_{t+1} \to \cdots$.

Tổng hợp cả hai nhánh (nhánh tương lai gói trong $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t+1}}$).



Gradient theo hidden state h_t

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} = \underbrace{\frac{1}{T}\frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{h}_t}}_{\text{nhánh output tại } t} + \underbrace{\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t+1}} \cdot \frac{\partial \boldsymbol{h}_{t+1}}{\partial \boldsymbol{h}_t}}_{\text{nhánh hồi quy về tương lai}} .$$

Mở từng nhánh bằng chain rule:

$$\frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{h}_t} = \frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \cdot \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{z}_t} \cdot \frac{\partial \boldsymbol{z}_t}{\partial \boldsymbol{h}_t} \qquad \text{(chuỗi } \boldsymbol{h}_t \to \boldsymbol{z}_t \to \boldsymbol{o}_t \to \ell\text{)}$$

$$= \boldsymbol{W}_{qh}^\top \left(\frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \odot \phi_o'(\boldsymbol{z}_t) \right),$$

$$\frac{\partial \boldsymbol{h}_{t+1}}{\partial \boldsymbol{h}_t} = \frac{\partial \phi_h(\boldsymbol{a}_{t+1})}{\partial \boldsymbol{a}_{t+1}} \cdot \frac{\partial \boldsymbol{a}_{t+1}}{\partial \boldsymbol{h}_t} = \operatorname{diag}\left(\phi_h'(\boldsymbol{a}_{t+1})\right) \boldsymbol{W}_{hh}^\top.$$

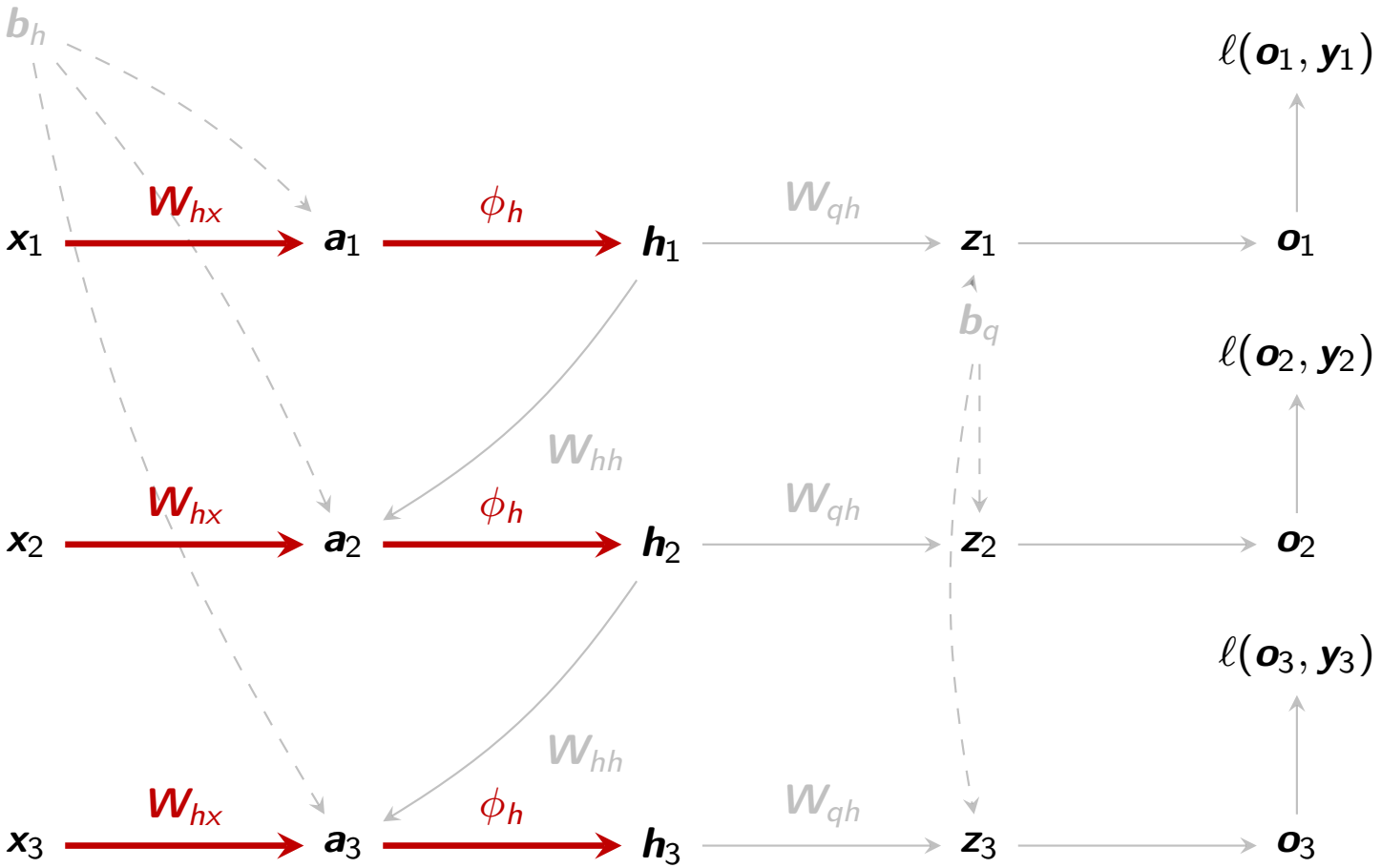
Ghép lại:

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} = \frac{1}{T} \boldsymbol{W}_{qh}^\top \left(\frac{\partial \ell}{\partial \boldsymbol{o}_t} \odot \phi_o'(\boldsymbol{z}_t) \right) + \boldsymbol{W}_{hh}^\top \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t+1}} \odot \phi_h'(\boldsymbol{a}_{t+1}) \right),$$

Đường đạo hàm cho W_{hx}

Chuỗi: $W_{hx} \to a_t \to h_t \to z_t \to o_t \to \ell_t$ tại mỗi timestep.

Cộng các đóng góp qua thời gian (do chia sẻ tham số) rồi lấy trung bình.



Gradient theo W_{hx}

Nhắc lại tiền kích hoạt & hidden:

$$\boldsymbol{a}_t = \boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h, \qquad \boldsymbol{h}_t = \phi_h(\boldsymbol{a}_t).$$

Với W_{hx} (input $\to$ hidden):

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{hx}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \cdot \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{a}_t} \cdot \frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{W}_{hx}} \qquad \text{(dùng trực tiếp } \tfrac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \text{ đã suy ra)}$$

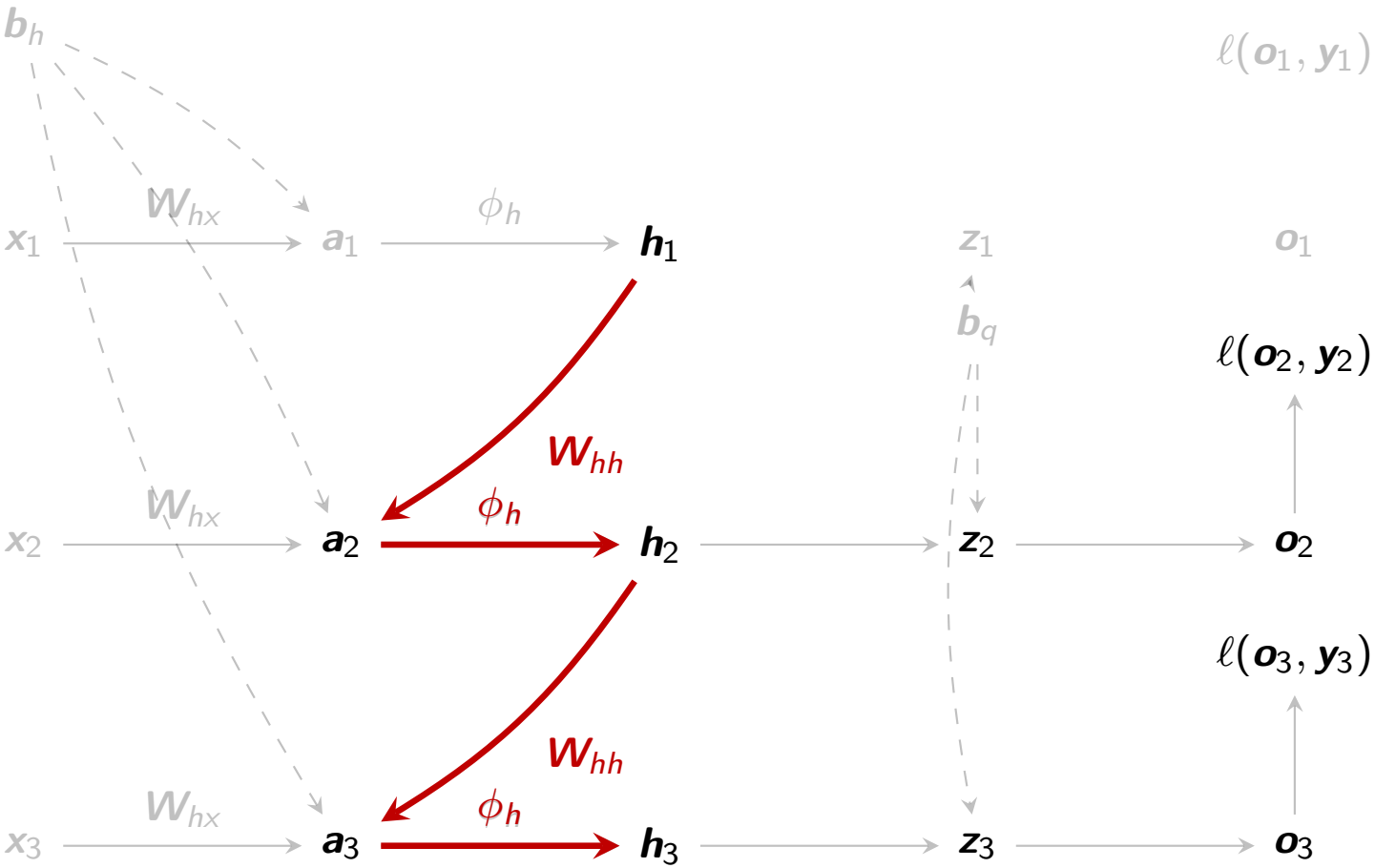
$$= \sum_{t=1}^{T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t) \right) \boldsymbol{x}_t^\top,$$

vì $\partial \boldsymbol{a}_t / \partial \boldsymbol{W}_{hx} = \boldsymbol{x}_t$. Lưu ý $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}$ đã bao gồm hệ số $1/T$ và các đóng góp hồi quy phía sau.

Đường đạo hàm cho W_{hh}

Chuỗi: $\boldsymbol{h}_{t-1} \xrightarrow{W_{hh}} \boldsymbol{a}_t \to \boldsymbol{h}_t \to \boldsymbol{z}_t \to \boldsymbol{o}_t \to \ell_t$.

Cộng dồn qua các timestep có kết nối hồi quy, rồi chia trung bình.



Gradient theo W_{hh}

Với $\boldsymbol{a}_t = \boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h$ suy ra $\partial \boldsymbol{a}_t / \partial \boldsymbol{W}_{hh} = \boldsymbol{h}_{t-1}$.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{hh}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \cdot \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{a}_t} \cdot \frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{W}_{hh}} \qquad \text{(chain rule, dùng } \tfrac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \text{ từ slide trước)}$$

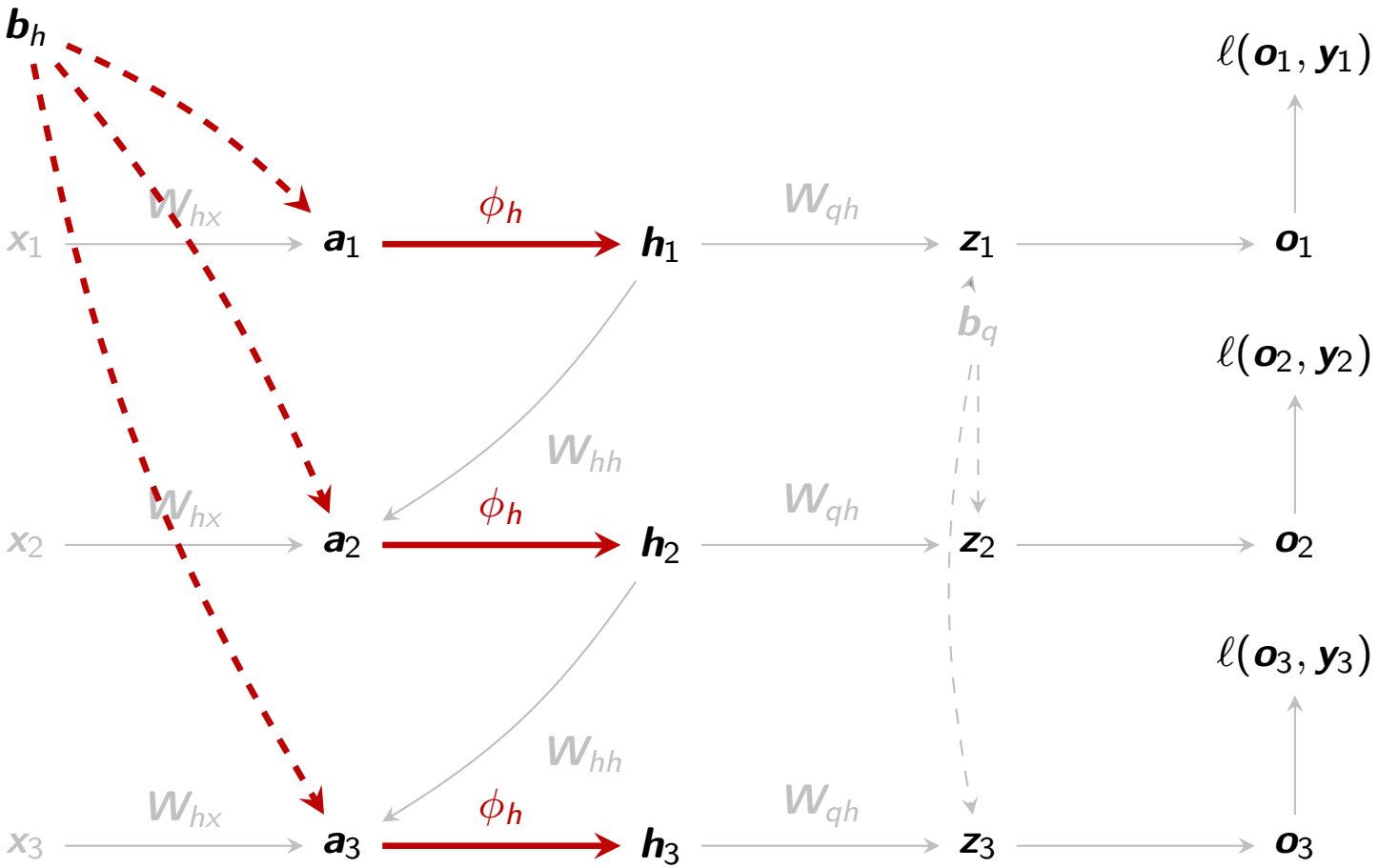
$$= \sum_{t=1}^{T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t) \right) \boldsymbol{h}_{t-1}^{\top},$$

vì $\partial \boldsymbol{a}_t / \partial \boldsymbol{W}_{hh} = \boldsymbol{h}_{t-1}$. Hệ số $1/T$ và ảnh hưởng từ các bước tương lai đã nằm trong $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}$.

Đường đạo hàm cho b_h

Nhánh: $\boldsymbol{b}_h \dashrightarrow \boldsymbol{a}_t \to \boldsymbol{h}_t \to \boldsymbol{z}_t \to \boldsymbol{o}_t \to \ell_t$ tại mỗi t.

Bias chia sẻ nên các đóng góp được cộng lại rồi chia T.



Gradient theo bias $\boldsymbol{b}_h$

Từ $\boldsymbol{a}_t = \boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h$ ta có $\partial \boldsymbol{a}_t / \partial \boldsymbol{b}_h = \boldsymbol{I}$.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_h} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \cdot \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{a}_t} \cdot \frac{\partial \boldsymbol{a}_t}{\partial \boldsymbol{b}_h} \qquad \text{(chain rule qua } \boldsymbol{a}_t\text{)}$$

$$= \sum_{t=1}^{T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t) \right) \qquad \text{(do } \partial \boldsymbol{a}_t / \partial \boldsymbol{b}_h = \boldsymbol{I}\text{)}.$$

Tương tự như $\boldsymbol{W}_{hx}$, hệ số $1/T$ và các ảnh hưởng lan ngược đã nằm trong $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}$.

Tóm tắt gradient các tham số

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_q} = \frac{1}{T}\sum_{t=1}^{T} \frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \odot \phi_o'(\boldsymbol{z}_t),$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{qh}} = \frac{1}{T}\sum_{t=1}^{T} \left(\frac{\partial \ell(\boldsymbol{o}_t, \boldsymbol{y}_t)}{\partial \boldsymbol{o}_t} \odot \phi_o'(\boldsymbol{z}_t)\right) \boldsymbol{h}_t^\top,$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{hx}} = \sum_{t=1}^{T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t)\right) \boldsymbol{x}_t^\top,$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{W}_{hh}} = \sum_{t=1}^{T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t)\right) \boldsymbol{h}_{t-1}^\top,$$

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{b}_h} = \sum_{t=1}^{T} \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \odot \phi_h'(\boldsymbol{a}_t)\right).$$

Điều kiện biên: $\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{T+1}} = 0.$

Demo: RNN dự đoán giá cổ phiếu

Bài toán & kiến trúc

Many-to-one: cửa sổ 20 giá $\rightarrow$ dự báo 1 giá tiếp theo.

Kiến trúc: 1 input, 16 hidden (tanh), 1 output tuyến tính.

Loss: MSE trên output cuối chuỗi; dữ liệu min-max fit trên train, dùng lại cho test.

Huấn luyện

300 epochs, lr $= 0.01$.

Gradient clipping $\tau = 5$ để tránh exploding.

Hiệu năng test

RMSE ≈ 136.39

MAE ≈ 135.67

Code forward RNN

```
1     def forward(self, x_seq, h0=None, return_cache: bool = False):
2         # Forward cho mot chuoi (T, input_size). Tra outputs (T,
              output_size) va h_T.
3         x_seq = np.asarray(x_seq, dtype=float)
4         if x_seq.ndim == 1:
5             x_seq = x_seq.reshape(-1, self.input_size)
6         T = x_seq.shape[0]
7         h_prev = np.zeros(self.hidden_size) if h0 is None else h0
8
9         hs = [h_prev]
10        pre_acts = []
11        outputs = []
12
13        for t in range(T):
14            a_t = x_seq[t].dot(self.W_hx) + h_prev.dot(self.W_hh)
                  + self.b_h
15            h_t = np.tanh(a_t)
16            o_t = h_t.dot(self.W_qh) + self.b_q
```

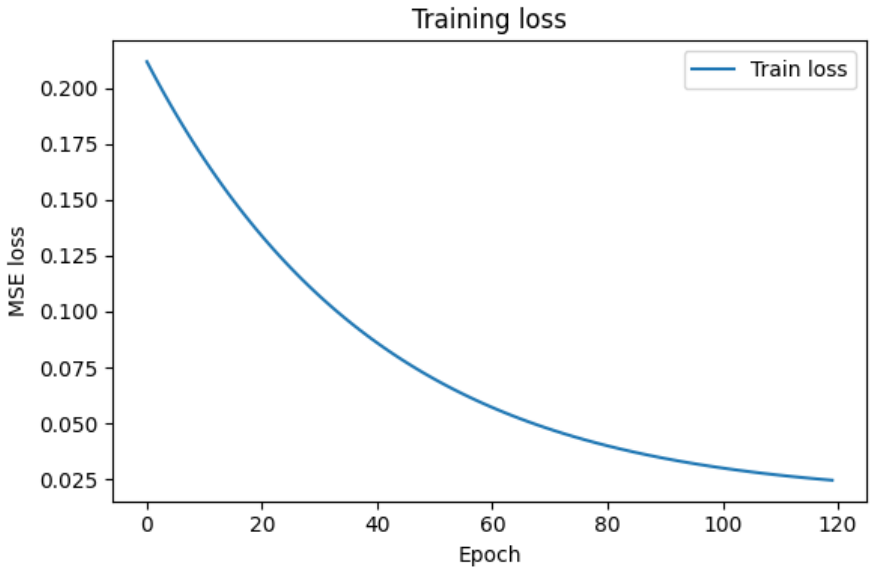
Code backward (1/2)

```
1     def backward(self, cache: dict, target: np.ndarray):
2         # BPTT cho loss 0.5*(o_T - y)^2. Chi dung output cuoi
              chuoi.
3         x_seq = cache["x_seq"]
4         hs = cache["hs"]
5         pre_acts = cache["pre_acts"]
6         outputs = cache["outputs"]
7
8         T = x_seq.shape[0]
9         target = np.asarray(target).reshape(-1)
10        o_T = outputs[-1].reshape(-1)
11
12        grads = {
13            "W_hx": np.zeros_like(self.W_hx),
14            "W_hh": np.zeros_like(self.W_hh),
15            "W_qh": np.zeros_like(self.W_qh),
16            "b_h": np.zeros_like(self.b_h),
17            "b_q": np.zeros_like(self.b_q),
18        }
19
20        dL_do = o_T - target  # derivative of 0.5*(o - y)^2
21        grads["W_qh"] += np.outer(hs[-1], dL_do)
```

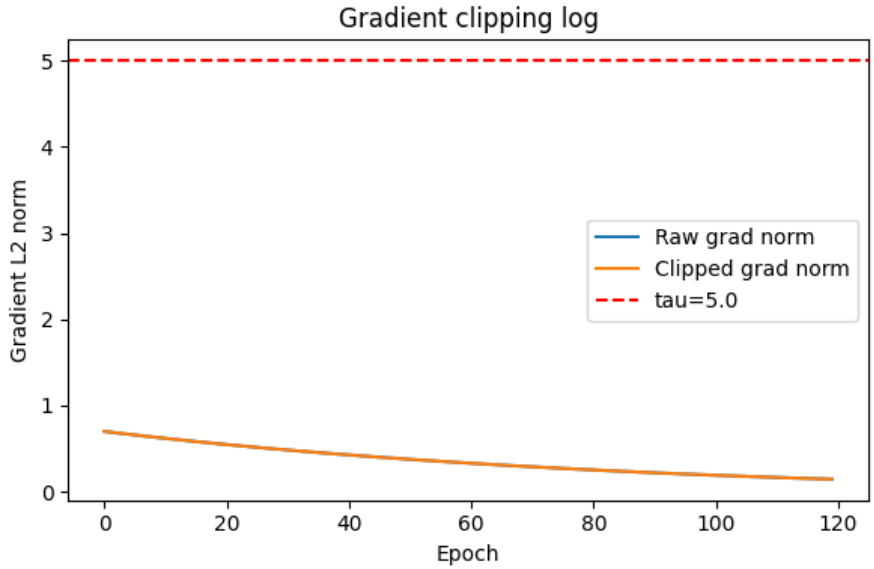
Code backward (2/2)

```
        grads["b_q"] += dL_do

        d_next = dL_do.dot(self.W_qh.T)

        for t in reversed(range(T)):
            h_t = hs[t + 1]
            h_prev = hs[t]
            a_t = pre_acts[t]
            x_t = x_seq[t]

            dh = d_next
            da = dh * (1.0 - h_t**2)   # tanh'

            grads["W_hx"] += np.outer(x_t, da)
            grads["W_hh"] += np.outer(h_prev, da)
            grads["b_h"] += da

            d_next = da.dot(self.W_hh.T)

        loss = 0.5 * np.mean((o_T - target) ** 2)
        return loss, grads
```

Training loss và norm gradient (đã clipping)



Loss giảm đều trong 300 epochs.

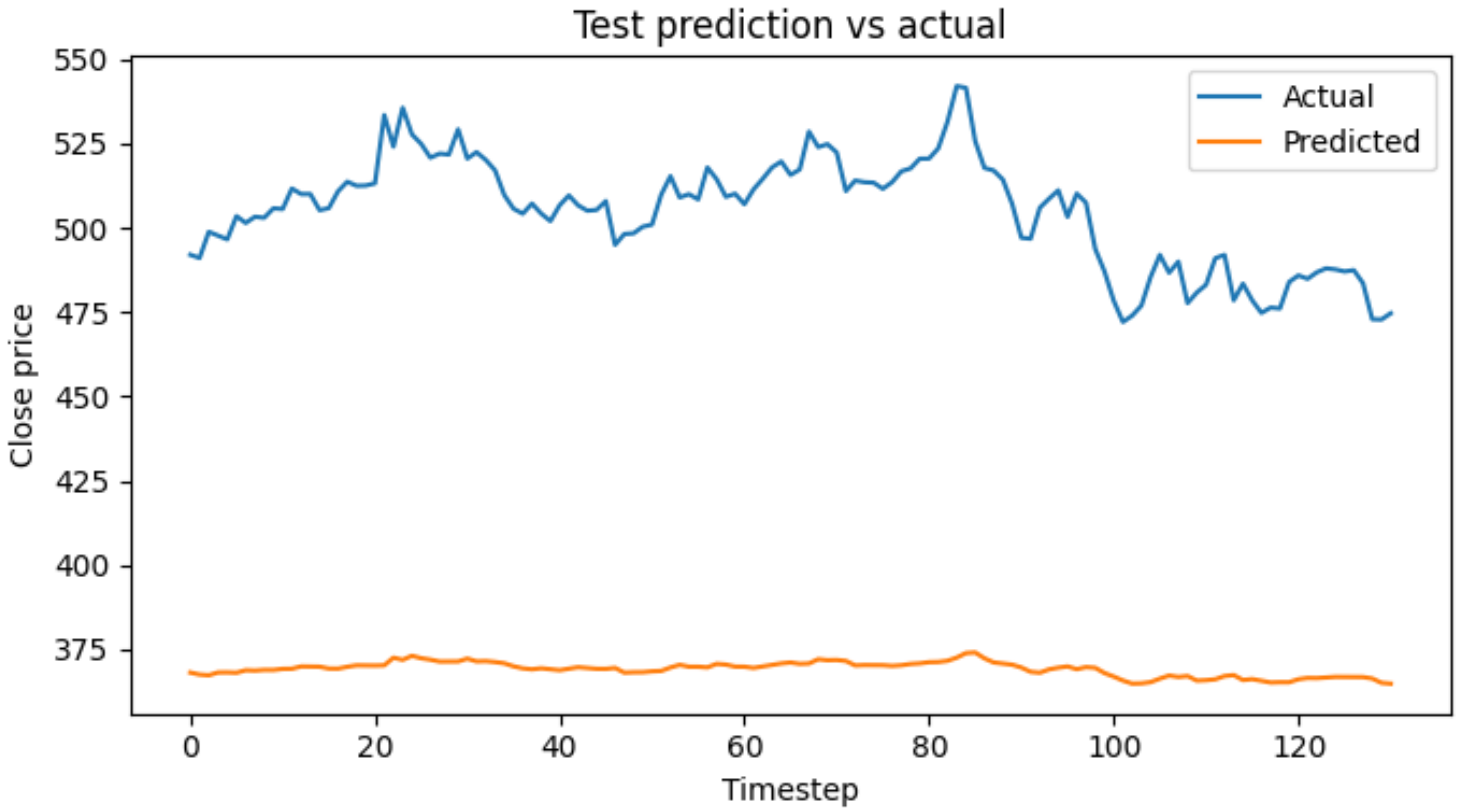


Clipping giữ gradient norm gần ngưỡng $\tau = 5$ và tránh exploding.

Dự báo trên tập test

Dùng tham số đã huấn luyện và scaler của train set để dự báo giá test.

Sai số: RMSE ≈ 136.39, MAE ≈ 135.67.



Mục lục chương 3

Nguồn gốc của vanishing và exploding gradients

Từ chương trước, gradient theo hidden state tại thời điểm t có dạng:

$$\frac{\partial L}{\partial h_t} = \sum_{k=t}^{T} (W_{hh}^{\top})^{k-t} W_{qh}^{\top} \frac{\partial L}{\partial o_k}. \tag{1}$$

Do đó, độ lớn của gradient phụ thuộc trực tiếp vào các lũy thừa của ma trận $W_{hh}^{\top}$.

Vấn đề cốt lõi:

- Gradient bị nhân lặp lại bởi cùng một ma trận qua nhiều bước thời gian.
- Hành vi của gradient được quyết định bởi W_{hh} (độ lớn/đặc trưng riêng của ma trận này).

Ước lượng độ lớn gradient

Lấy chuẩn hai vế:

$$\left\| \frac{\partial L}{\partial h_t} \right\| \leq \sum_{k=t}^{T} \left\| (W_{hh}^{\top})^{k-t} \right\| \left\| W_{qh}^{\top} \right\| \left\| \frac{\partial L}{\partial o_k} \right\|. \tag{2}$$

Với chuẩn ma trận dưới chuẩn vector:

$$\left\| (W_{hh}^{\top})^{k-t} \right\| \leq \| W_{hh} \|^{k-t}. \tag{3}$$

Do đó:

$$\left\| \frac{\partial L}{\partial h_t} \right\| \lesssim \sum_{k=t}^{T} \| W_{hh} \|^{k-t}. \tag{4}$$

Điều kiện vanishing và exploding gradients

Xét giới hạn khi độ dài chuỗi tăng ($T \to \infty$):

Nếu $\|W_{hh}\| < 1$:

$$\|W_{hh}\|^{k-t} \to 0 \quad \Rightarrow \quad \text{gradient vanishing.}$$

Nếu $\|W_{hh}\| > 1$:

$$\|W_{hh}\|^{k-t} \to \infty \quad \Rightarrow \quad \text{gradient exploding.}$$

Nếu $\|W_{hh}\| \approx 1$:

gradient được duy trì ổn định.

Trong thực hành, điều kiện này tương đương với bán kính phổ (spectral radius) của W_{hh}.

Tóm tắt vấn đề

Vanishing và exploding gradients là hệ quả trực tiếp của việc:

- Nhân lặp cùng một ma trận W_{hh} qua nhiều bước thời gian.
- Gradient là tổng của các chuỗi lũy thừa ma trận.

Do đó, việc huấn luyện RNN chuỗi dài gặp khó khăn ngay cả trong trường hợp tuyến tính đơn giản.

Giải pháp 1: Gradient Clipping

Gradient clipping giới hạn độ lớn gradient trong quá trình cập nhật.

Cụ thể:

$$g \leftarrow \frac{g}{\max\left(1, \frac{\|g\|}{\tau}\right)}, \tag{5}$$

trong đó g là gradient và τ là ngưỡng.

Đặc điểm:

- Không loại bỏ nguyên nhân gốc rễ.
- Ngăn gradient exploding trong thực tế.
- Dễ cài đặt, được dùng phổ biến.

Giải pháp 2: Long Short-Term Memory (LSTM)

LSTM thay đổi kiến trúc RNN bằng cách:

- Tạo đường truyền gradient gần như tuyến tính theo thời gian.
- Tránh nhân lặp trực tiếp bởi cùng một ma trận.

Trạng thái cell c_t được cập nhật dưới dạng:

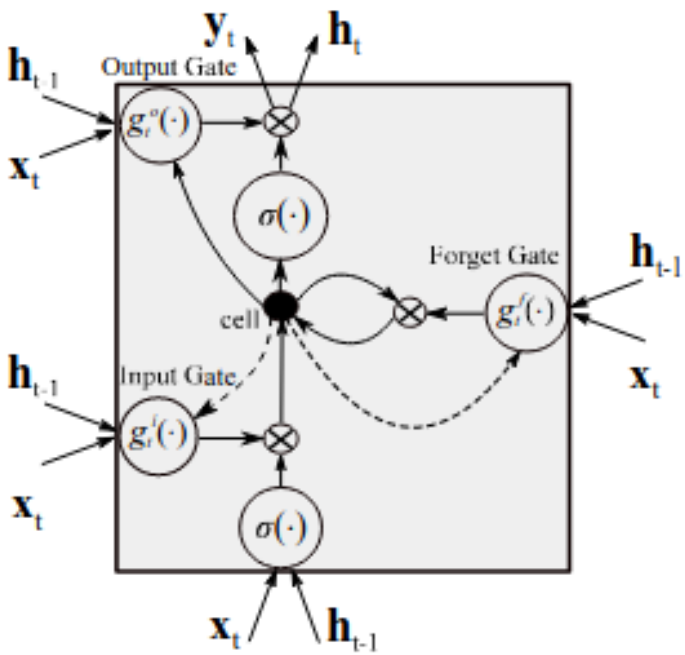
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t, \tag{6}$$

trong đó f_t là forget gate.

Gradient có thể đi xuyên qua nhiều bước thời gian mà không bị vanishing hoặc exploding nghiêm trọng.

Minh họa Gradient Cliping và LSTM

$$
\begin{aligned}
&\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta} \\
&\textbf{if } \|\hat{\mathbf{g}}\| \geq threshold \textbf{ then} \\
&\quad \hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|} \hat{\mathbf{g}} \\
&\textbf{end if}
\end{aligned}
$$



Kết quả thực nghiệm vanishing and exploding