وزارت علوم، تحقیقات و فناوری
دانشگاه تحصیلات تکمیلی علوم پایه
گاوازنگ، زنجان

دانشکده علوم رایانه و فناوری اطلاعات

درس: یادگیری ژرف (Deep Learning)
تمرین شماره ۲

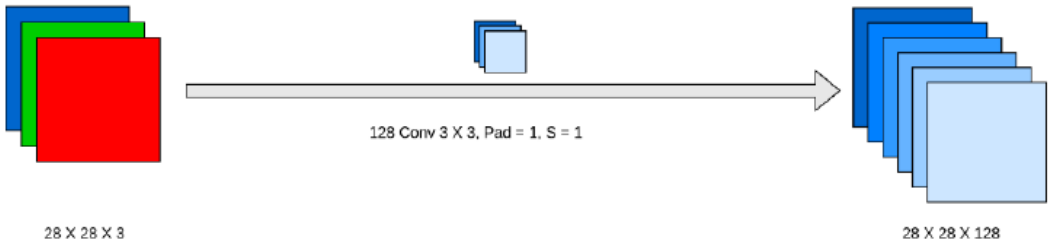
استاد: دکتر رزاقی

دانشجو: امیرحسین صفری

شماره دانشجویی: ۱۴۰۱۴۱۲۱

بهار ۱۴۰۲

(A) در شکل ۱، یک پیمانه از شبکه MobileNet-V1 با یک لایه از شبکه CNN معمولی نمایش داده شده است.در هر روش تعداد پارامترها را محاسبه کنید.



پارامترها که به طور کلی وزنه هایی هستند که در طول تمرین یاد می گیرند . آن‌ها ماتریس‌های وزنی هستند که به قدرت پیش‌بینی مدل کمک می‌کنند، که در طول فرآیند back propagation تغییر می‌کنند. فرمول محاسبه‌ی تعداد آنها برای لایه‌های کانولوشنی به صورت زیر می‌باشد:

follows: ((shape of width of the filter * shape of height of the filter * number of filters in the previous layer+1)*number of filters). Where

که به طور خلاصه برابر است با:

Don't forget the bias term for each of the filter. Number of parameters in a CONV layer would be : ((m * n * d)+1)* k), added 1 because of the

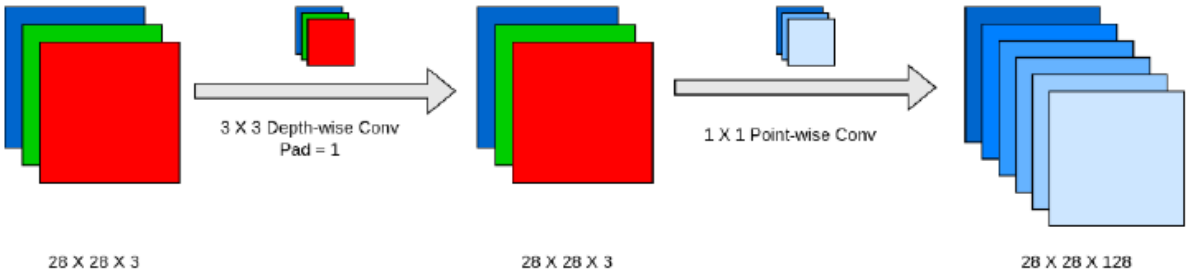
که در آن m عرض فیلتر، n طول(ارتفاع) فیلتر، d تعداد فیلترها(نقشه‌ی ویژگی یا feature map)ی لایه‌ی قبل و k تعداد فیلترها(feature map) در لایه‌ی کنونی است.

لذا برای شکل بالا که یک کانولوشن ساده است محاسبه می‌کنیم:

((۳ * ۳ * ۳) + ۱) * ۱۲۸ = ۳۵۸۴

برای حالتی که بایاس را در نظر نگرفته و پدینگ را در نظر می گیریم:

(((۳ * ۳) + ۱) * ۳) * ۱۲۸ = ۳۸۴۰



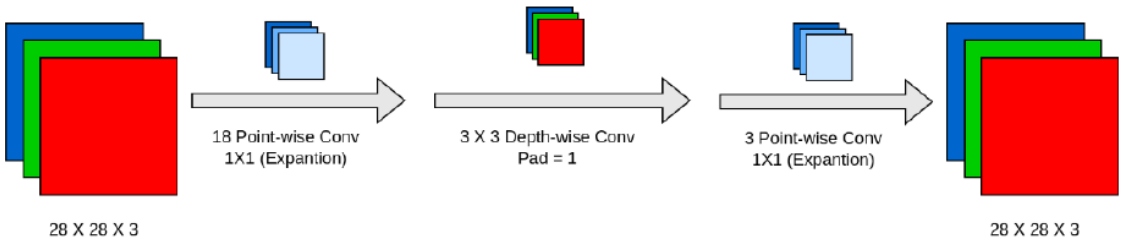
برای شکل بالا داریم:

[((۳ * ۳ * ۳) + ۱) * ۳] + [((۱ * ۱ * ۳) + ۱) * ۱۲۸] = ۵۹۶

برای حالتی که بایاس را در نظر نگرفته و پدینگ را در نظر می گیریم:

[((۳ * ۳) + ۱) * ۳] + [((۱ * ۱ * ۳) + ۱) * ۱۲۸] = ۵۴۲

(B) در شبکه MobileNet-V2 از دو فیلتر کانولوشن 1 × 1 است.شمای کلی معماری استفاده شده از این شبکه در شکل ۲نشان داده شده است.با محاسبه تعداد پارامتر ها برای این شبکه، این شبکه را با شبکه MobileNet-V1 مقایسه نمایید.



شکل ۲: لایه از شبکه MobileNet-V2

با توجه به فرمول در بخش (A) داریم:

[((۱ * ۱ * ۳) + ۱) * ۱۸] + [((۳ * ۳ * ۱۸) + ۱) * ۱۸] + [((۱ * ۱ * ۱۸) + ۱) * ۳] = ۳۰۶۳

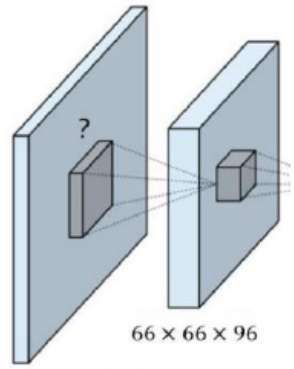
برای حالتی که بایاس را در نظر نگرفته و پدینگ را در نظر می گیریم:

[(۱ * ۱ * ۳) * ۱۸] + [((۳ * ۳) + ۱) * ۱۸] * ۱۸] + [((۱ * ۱ * ۱۸)) * ۳] = ۳۳۴۸

(قابل ذکر است که ابعاد خروجی کانولوشن میانی 3 * 3 depth-wise برابر با ۲۸ * ۲۸ * ۱۸ است. همچنین ابعاد خروجی کانولوشن اول برابر با ۲۸ * ۲۸ * ۱۸ و ابعاد خروجی کانولوشن آخر برابر با ۲۸ * ۲۸ * ۳ است.)

با توجه به اینکه تعداد پارامترها در شبکه‌ی MobileNet-V1 برابر با ۵۹۶ و تعداد پارامترها در شبکه‌ی MobileNet-V2 برابر با ۳۰۶۳ می‌باشد،لذا می‌توان نتیجه گرفت که پیچیدگی محاسباتی شبکه‌ی MobileNet-V2 از MobileNet-V1 بیشتر می‌باشد (بیش از ۵ برابر).

(A) با توجه به ابعاد ورودی و خروجی نشان داده شده در شکل زیر، سایز کرنل مورد استفاده در این عملیات کانولوشی را بدست آورید. لازم به ذکر است که ابعاد مشخص شده در شکل زیر، با احتساب **zero-padding** داده شده اند.



با توجه به فرمول زیر به محاسبه‌ی کرنل می‌پردازیم:

you can use this formula `[(W-K+2P)/S]+1`.

- W is the input volume - in your case 128
- K is the Kernel size - in your case 5
- P is the padding - in your case 0 i believe
- S is the stride - which you have not provided.

که در اینجا W برابر با ۲۰۵، P برابر با صفر، S برابر با ۳ است؛ داریم:

$$[(۲۰۵ - k + ۲ * 0) / ۳] + ۱ = ۶۶$$
$$k = ۱۰$$

(B) تعداد پارامتر های قابل آموزش یا **Learnable** موجود در لایه کانولوشـــنی را تعیین نمایید.(راهنمایی: به پارامتر بایاس در هر کرنل نیز در محاسبات خود داشته باشید.)

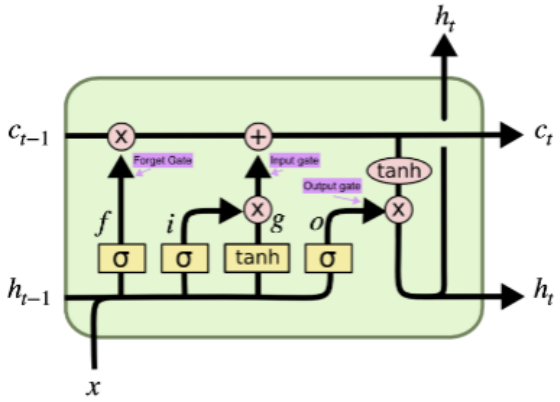
با توجه به فرمول سوال ۱ قسمت (A) داریم:

((۱۰ * ۱۰ * ۱۰) + ۱) * ۹۶ = ۹۶۰۹۶

(C) تعداد عملیات ضرب مورد نیاز برای بدست آوردن خروجی را محاسبه کنید.(ضرب های در صفر را نیز در شمارش تعداد ضرب ها لحاظ نمایید.)

تعداد حرکت کرنل در یک ردیف ۶۵ = ۳ / (۱۰ - ۲۰۵)
تعداد کل حرکت های کرنل روی ماتریس ورودی(با حساب کردن بایاس) (۱ + ۶۵) * (۱ + ۶۵) = ۴۳۵۶
تعداد ضرب‌های کل یک کرنل در یک فیلتر ۴۳۵۶ * (۱۰ * ۱۰ * ۱۰) = ۴۳۵۶۰۰۰
تعداد کل ضرب‌های فیلترها (تعداد کل) ۴۱۸۱۷۶۰۰۰ = ۹۶ * ۴۳۵۶۰۰۰

در این سوال می‌خواهیم نحوه مشتق گیری خطای پس از انتشار را در هر یک از سلول های شبکه های LSTM بررسی کنیم.
سلول LSTM زیر را در نظر بگیرید:



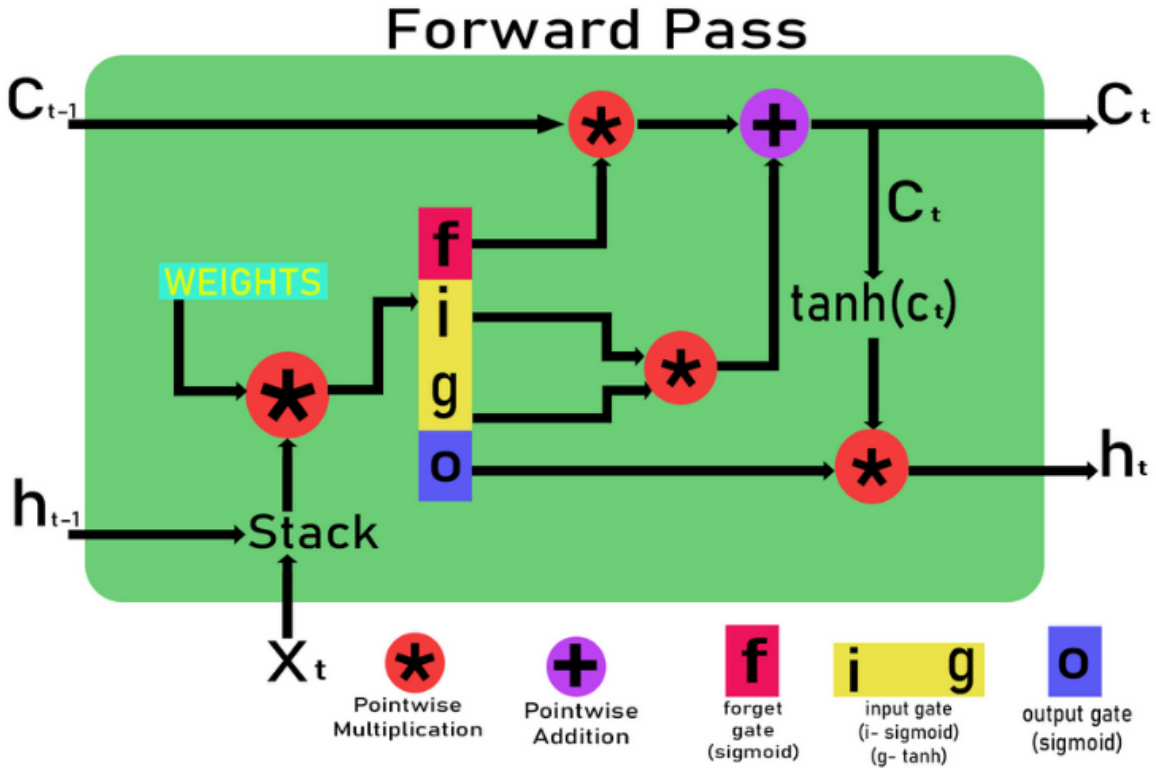
برای سلول نشان داده شده، مربوط به خروجی Forget gate، Input gate، Output gate، c_t و h_t را بنویسید.وزن گیت های مختلف به صورت زیر است:

$$Input\ gate: w_{xi}.w_{xg}.b_i.w_{hj}.w_g.b_g$$

$$Forget\ gate: w_{xf}.b_f.w_{hf}$$

$$Output\ gate: w_{xo}.b_0.w_{ho}$$

می‌خواهیم نحوه‌ی مشتق گیری خطای پس از انتشار را در هر یک از سلول‌های LSTM بررسی کنیم. همانطور که از نام آن پیداست، پس از انتشار(backpropagation) در طول زمان شبیه به پس از انتشار در DNN (شبکه عصبی عمیق deep neural network) است، اما به دلیل وابستگی زمان در RNN و LSTM، باید قانون زنجیره را با وابستگی زمانی اعمال کنیم.



فرض می‌کنیم که ورودی در زمان t در سلول LSTM، xt باشد. همچنین حالت سلول از زمان t-1 و t به صورت ct-1 و ct باشد و خروجی برای زمان t-1 و t به صورت ht-1 و ht باشد. مقدار اولیه ct و ht در t = 0 صفر خواهد بود.

- مرحله 1: مقدار دهی اولیه وزن ها

وزن برای گیت های مختلف عبارتند از:

گیت ورودی: $w_{xi}, w_{xg}, b_i, w_{hj}, w_g , b_g$

گیت فراموشی: w_{xf}, b_f, w_{hf}

گیت خروجی: w_{xo}, b_o, w_{ho}

- مرحله 2: عبور از گیت های مختلف

ورودی ها: xt و ct-1، ht-i به سلول LSTM داده می شود.

عبور از گیت ورودی:

$$Z_g = w_{xg} * x + w_{hg} * h_{t-1} + b_g$$
$$g = \tanh(Z_g)$$
$$Z_j = w_{xi} * x + w_{hi} * h_{t-1} + b_i$$
$$i = \text{sigmoid}(Z_i)$$

Input_gate_out = g*i

عبور از گیت فراموشی:

$Z_f = w_{xf} * x + w_{hf} * h_{t-1} + b_f$
$f = sigmoid(Z_f)$

Forget_gate_out = f

عبور از گیت خروجی:

$Z_o = w_{xo} * x + w_{ho} * h_{t-1} + b_o$
$o = sigmoid(z_O)$

Out_gate_out = o

- مرحله 3: محاسبه خروجی ht و وضعیت سلول فعلی ct

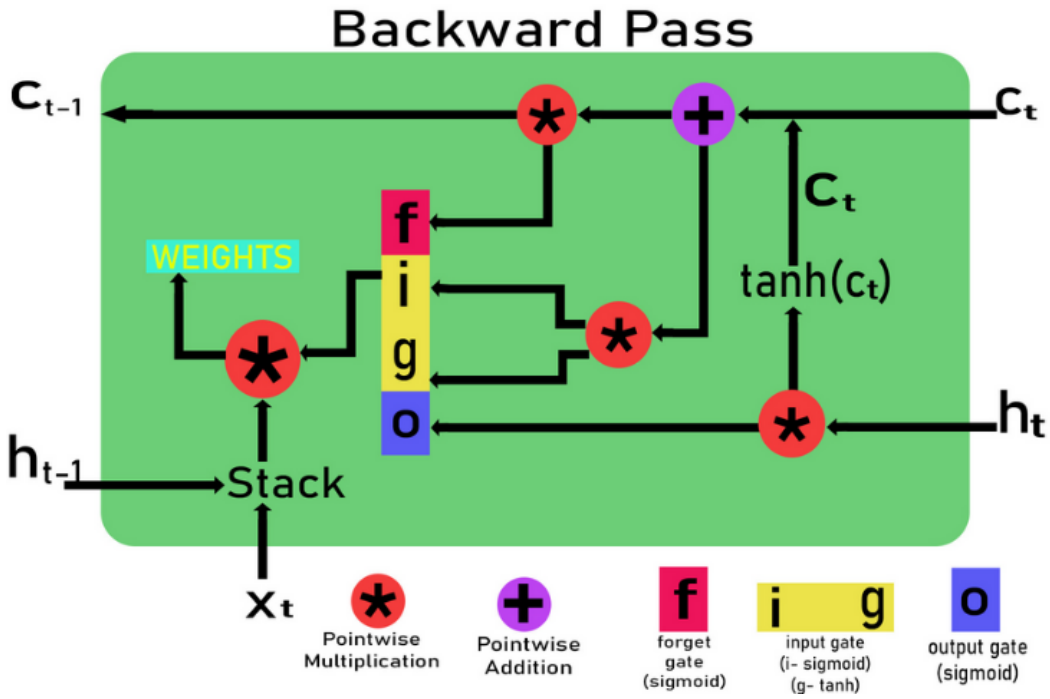
محاسبه وضعیت سلول فعلی ct:

$c_t = (c_{t-1} * forget\_gate\_out) + input\_gate\_out$

محاسبه گیت خروجی ht:

$h_t = out\_gate\_out * tanh(ct)$

- مرحله 4: محاسبه گرادیان از طریق پس از انتشار(back propagation) در طول زمان در زمان t با استفاده از قانون زنجیره



فرض می‌کنیم، گرادیان عبوری از سلول بالا به صورت زیر باشد:

$E\_delta = dE/dh_t$

اگر از MSE (میانگین مربع خطا) برای خطا استفاده می‌کنیم، داریم:

$E\_delta=(y-h(x))$

در اینجا y مقدار اصلی h(x) و مقدار پیش بینی شده است.

گرادیان با توجه به گیت خروجی:

$$dE/do = (dE/dh_t) * (dh_t/do) = E\_delta * (dh_t / do)$$
$$dE/do = E\_delta * \tanh(c_t)$$

گرادیان با توجه به c_t:

$$dE/dc_t = (dE / dh_t)*(dh_t/dc_t) = E\_delta *(dh_t/dc_t)$$
$$dE/dc_t = E\_delta * o * (1-\tanh^2(c_t))$$

گرادیان با توجه به گیت ورودی dE/di، dE/dg

$$dE/di = (dE/di) * (dc_t / di)$$
$$dE/di = E\_delta * o * (1-\tanh^2(c_t)) * g$$

به طور مشابه:

$$dE/dg = E\_delta * o * (1-\tanh^2(c_t)) * i$$

گرادیان با توجه به گیت فراموشی:

$$dE/df = E\_delta * (dE/dc_t) * (dc_t / dt)\ t$$
$$dE/df = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1}$$

گرادیان با توجه به c_{t-1}:

$$dE/dc_t = E\_delta * (dE/dc_t) * (dc_t / dc_{t-1})$$
$$dE/dc_t = E\_delta * o * (1-\tanh^2(c_t)) * f$$

گرادیان با توجه به وزن گیت خروجی:

$$dE/dw_{xo} = dE/d_o * (d_o/dw_{xo}) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o) * x_t$$
$$dE/dw_{ho} = dE/do * (do/dw_{ho}) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o) * h_{t-1}$$
$$dE/db_o = dE/do * (do/db_o) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)$$

گرادیان با توجه به وزنه های گیت فراموشی:

$$dE/dw_{xf} = dE/df * (df/dw_{xf}) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * x_t$$
$$dE/dw_{hf} = dE/df * (df/dw_{hf}) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f) * h_{t-1}$$
$$dE/db_o = dE/df * (df/db_o) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)$$

گرادیان با توجه به وزن گیت ورودی:

$$dE/dw_{xi} = dE/di * (di/dw_{xi}) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * x_t$$
$$dE/dw_{hi} = dE/di * (di/dw_{hi}) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i) * h_{t-1}$$
$$dE/db_i = dE/di * (di/db_i) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)$$

$$dE/dw_{xg} = dE/dg * (dg/dw_{xg}) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g)) * x_t$$
$$dE/dw_{hg} = dE/dg * (dg/dw_{hg}) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g)) * h_{t-1}$$
$$dE/db_g = dE/dg * (dg/db_g) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g))$$

در نهایت گرادیان های مرتبط با وزن ها عبارتند از:

- $dE/dw_{xo} = dE/d_o * (d_o/dw_{xo}) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)) * x_t$
- $dE/dw_{ho} = dE/do * (do/dw_{ho}) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)) * h_{t-1}$
- $dE/db_o = dE/do * (do/db_o) = E\_delta * \tanh(c_t) * sigmoid(z_o) * (1-sigmoid(z_o)$
- $dE/dw_{xf} = dE/df * (df/dw_{xf}) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)) * x_t$
- $dE/dw_{hf} = dE/df * (df/dw_{hf}) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)) * h_{t-1}$
- $dE/db_o = dE/df * (df/db_o) = E\_delta * o * (1-\tanh^2(c_t)) * c_{t-1} * sigmoid(z_f) * (1-sigmoid(z_f)$
- $dE/dw_{xi} = dE/di * (di/dw_{xi}) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)) * x_t$
- $dE/dw_{hi} = dE/di * (di/dw_{hi}) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)) * h_{t-1}$
- $dE/db_i = dE/di * (di/db_i) = E\_delta * o * (1-\tanh^2(c_t)) * g * sigmoid(z_i) * (1-sigmoid(z_i)$
- $dE/dw_{xg} = dE/dg * (dg/dw_{xg}) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g)) * x_t$
- $dE/dw_{hg} = dE/dg * (dg/dw_{hg}) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g)) * h_{t-1}$
- $dE/db_g = dE/dg * (dg/db_g) = E\_delta * o * (1-\tanh^2(c_t)) * i * (1-\tanh^2(z_g))$

با استفاده از همه گرادیان‌ها، می‌توانیم به راحتی وزن‌های مرتبط با دروازه ورودی، دروازه خروجی و دروازه فراموشی را به‌روزرسانی کنیم.