УДК 004.8:004.032.26

Є.В. Бодянський, С.О. Костюк

НЕЙРОН НА ОСНОВІ АДАПТИВНОГО НЕЧІТКОГО ПЕРЕТВОРЕННЯ ДЛЯ СУЧАСНИХ МОДЕЛЕЙ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

Бодянський Євгеній Володимирович

Харківський національний університет радіоелектроніки,
orcid: 0000-0001-5418-2143
yevgeniy.bodyanskiy@nure.ua

Костюк Сергій Олександрович

Харківський національний університет радіоелектроніки,
orcid: 0000-0003-4196-2524
serhii.kostiuk@nure.ua

Зі зростанням об’ємів даних для обробки та з ускладненням задач з обробки даних науковці та спеціалісти з індустрії поступаються складністю та швидкістю моделей штучних нейронних мереж (ШНМ) на користь покращення їх апроксимуючих здатностей. Підвищення складності моделей, зокрема збільшення глибини та кількості шарів, призводить до небажаних ефектів, таких як зникаючі та вибухаючі градієнти. Комерційні моделі ШНМ часто використовують кусково-лінійні активаційні функції типу ReLU для уникнення обчислювальних складнощів та прискорення навчання. Хоча кусково-лінійні активаційні функції і доказали ефективність у комерційних моделях на прикладі згорткових моделей (Convolutional Neural Networks — CNN), для класифікації зображень вони, як правило, мають фіксовану форму, що обмежує здатність моделі до оптимізації та адаптування до поточної задачі. Запропоновано адаптивну кусково-лінійну активаційну функцію (Adaptive Piece-Wise Activation — APWA) як адаптивну альтернативу для фіксованих кусково-лінійних активацій. Основою APWA-функції є адаптивне нечітке перетворення вхідного сигналу, реалізоване множиною функцій належності з адаптивними параметрами підсилення вихідного сигналу. Як і кусково-лінійні активаційні функції, APWA позбавлена ефектів вибухаючого та зникаючого градієнтів, а також відносно проста в обчисленні, що зменшує тривалість навчання та сприяє прямому поширенню в мережах з нейронами на основі APWA. Показано ефективність нейронів та моделей на основі APWA на прикладі двох різних наборів даних для класифікації зображень, а також двох моделей різного рівня складності. Моделі з APWA адаптують форму активаційних функцій у процесі навчання, що покращує точність класифікації порівняно з базовими моделями, які не є адаптивними.

Ключові слова: модель нейронної мережі, адаптивне нечітке перетворення, активаційна функція.

Вступ

Здатність до навчання на основі даних у комбінації з властивостями універсальної апроксимації сприяла широкому поширенню штучних нейронних мереж (ШНМ) у сучасних системах обробки інформації [1]. Основою початкових моделей ШНМ є елементарний персептрон Ф. Розенблата [2], де властивості універсального апроксиматора забезпечуються суперпозицією сигмоїдальних активаційних функцій [3, 4], що стоять на виході кожного з нейронів моделі. Глибокі нейронні мережі (Deep Neural Networks — DNNs) розроблені на основі ідей традиційних ШНМ, але апроксимаційні здатності глибоких мереж розширені шляхом ускладнення архітектури мережі та збільшення кількості шарів і, відповідно, налаштовних параметрів [5]. Збільшення кількості шарів та параметрів мережі призводить до значних обчислювальних труднощів в DNN, а саме, ефектів зникаючого та вибухаючого градієнтів, які сповільнюють процес навчання та, в крайніх випадках, призводять до передчасної зупинки прогресу в процесі навчання. Для подолання цих проблем новітні глибокі моделі, такі як GPT-3 [6] та CoAtNet [7], використовують надлишкові зв'язки, складні механізми уваги, а також лінійні та випрямляючі активаційні функції з покращеними властивостями прямого поширення інформації.

Сигмоїдальні функції часто замінюються на кусково-задані лінійні активаційні функції для подолання обчислювальних складнощів в DNN. Кусково-задані функції, такі як випрямлений лінійний вузол (ReLU) [8] та його модифікації, не призводять до ефектів зникаючого та вибухаючого градієнтів через властивості їх похідних функцій. Одночасно такі кусково-задані активаційні функції не задовольняють умови теореми Дж. Цибенко [3], по факту втілюючи кусково-лінійну апроксимацію з обмеженою точністю, а досягнення бажаної якості моделі потребує збільшення загальної кількості шарів, нейронів та налаштовних параметрів. Як альтернатива можливе збільшення апроксимаційної здатності мережі шляхом заміни існуючих активаційних функцій такими, що здатні адаптуватись до задачі та змінювати форму в процесі навчання подібно до синаптичних ваг.

Водночас нейрони з подвійним нечітким перетворенням [9] мають відносно складну структуру, що вимагає додаткових обчислень та сповільнює процес навчання. Таким чином, є сенс звернутись до більш простих структур.

Метою дослідження є розробка адаптивної активаційної функції на основі нечіткого перетворення, що збільшує апроксимаційну здатність мережі порівняно з функціями типу ReLU, але має більш просту структуру щодо подвійного нечіткого перетворення.

У даній роботі пропонуємо адаптивну кусково-задану активаційну функцію (Adaptive Piece-Wise Activation — APWA) як адаптивну заміну для фіксованих кусково-заданих активаційних функцій. Для оцінки ефективності нейронів з APWA використовуємо існуючі архітектури ШНМ, порівнюючи мережі ReLU та APWA із задачами класифікації зображень.

1. Принцип роботи, структура та процес навчання нейрона з APWA

Робота нейрона з APWA заснована на принципі нечіткого перетворення вхідного сигналу. Структура нейрона включає в себе вузли та налаштовні параметри стандартного нейрона типу Adaline [10], а також елементи неофаззі нейрона (NFN) [11] як нелінійне перетворення, що має відносно високу швидкість, а також придатне для обробки інформації в потоковому режимі [12].

Нейрон з APWA має двошарову структуру (рис. 1). Перший шар містить набір налаштовних синаптичних ваг $w_j = (w_{j0}, w_{j1}, \ldots, w_{jn})$, що перетворюють вектор вхідних значень $x(t) = (1, x_1(t), \ldots, x_n(t))^{\mathrm{T}}$ розмірністю $(n + 1)$ у внутрішній сигнал

активації $z_j(t)$:

$$z_j(t) = \sum_{i=0}^{n} w_{ji} x_i(t) = w_j^{\mathrm{T}} x(t)$$

де $(n+1)$ відповідає розмірності вхідного вектора, $j = 1, 2, \ldots, h$ — індекс нейрона в шарі мережі, h — кількість нейронів у шарі, $t = 1, 2, \ldots, \tau$ — індекс чергового екземпляра сигналу в потоці вхідних даних, τ — число вхідних екземплярів.

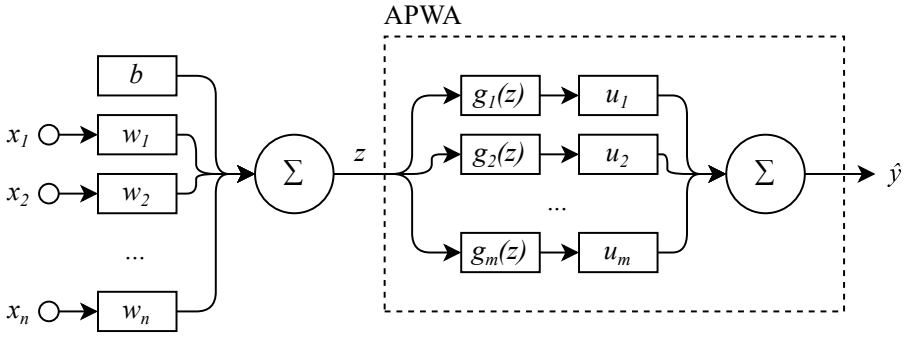

Рис. 1

Другий шар — нелінійний синапс APWA, що виконує нечітке перетворення внутрішнього сигналу активації $z_j(t)$. Він містить у собі m функцій належності $g_j(t) = (g_{j1}(z_j(t)), \ldots, g_{jm}(z_j(t)))^{\mathrm{T}}$, вихід яких помножується на m налаштовних ваг $u_j = (u_{j1}, \ldots, u_{jm})^{\mathrm{T}}$ та складається у сигнал:

$$\hat{y}_j(t) = \sum_{k=1}^{m} u_{jk} g_{jk}(z_j(t)) = u_j^{\mathrm{T}} z_j(t) \tag{1}$$

або у скалярну форму:

$$\hat{y}_j(t) = \varphi_j(z_j(t)) = \sum_{k=1}^{m} u_{jk} g_{jk}\left(\sum_{i=1}^{n} w_{ji} x_i(t)\right).$$

З властивостей нечіткого перетворення [13] видно, що таке перетворення дозволяє апроксимувати існуючі активаційні функції загального призначення на заданому інтервалі $z_{j,\min} \le z_j(t) \le z_{j,\max}$ з довільним рівнем точності, а також синтезувати нові шляхом налаштування вектора ваг u_j для оптимізації цільової функції $J(t)$.

Як нелінійні функції належності використовуємо набір стандартних рівномірно розподілених трикутних функцій такої форми:

$$g_{jk}(z_j) = \begin{cases} \frac{z_j - c_{j,k-1}}{c_{jk} - c_{j,k-1}} = \frac{z_j - c_{j,k-1}}{\Delta_c}, & z_j \in \left[c_{j,k-1}, c_{jk}\right], \\ \frac{c_{j,k+1} - z_j}{c_{j,k+1} - c_{jk}} = \frac{c_{j,k+1} - z_j}{\Delta_c}, & z_j \in \left[c_{jk}, c_{j,k+1}\right], \\ 0, & z_j \notin \left[c_{j,k-1}, c_{j,k+1}\right]. \end{cases} \tag{2}$$

Таким чином, якщо сигнал внутрішньої активації належить до інтервалу визначення функції APWA $z_{j,\min} \le z_j(t) \le z_{j,\max}$, то $c_{j1} = z_{j,\min}$, $c_{jm} = z_{j,\max}$, а також $\Delta_c = \frac{c_j - c_{jk}}{m-1}$, де c_{jk} — центр координат по вхідному сигналу функції $g_{jk}(z_j)$.

Оскільки функції належності (2) реалізують рівномірне розбиття вхідного простору на інтервалі визначення, тобто для будь-яких сусідніх функцій дійсним є

$$g_{j,k-1}(z_j) + g_{jk}(z_j) = g_{jk}(z_j) + g_{j,k+1}(z_j) = 1,$$

то нелінійне перетворення (1) для $z_j(t) \in \left[c_{jk}, c_{j,k+1}\right]$ можна записати

$$\hat{y}(t) = \sum_{k=1}^{m} u_{jk} g_{jk}(z_j(t)) = u_{jk} g_{jk}(z_j(t)) + u_{j,k+1} g_{j,k+1}(z_j(t)) =$$

$$= \frac{c_{j,k+1}u_{jk} - c_{jk}u_{j,k+1}}{\Delta_c} + \frac{\left(u_{j,k+1} - u_{jk}\right) z_j(t)}{\Delta_c} =$$

$$= \nu_{j0,k,k+1} + \nu_{j1,k,k+1} z_j(t) = \nu_{j0,k,k+1} + \nu_{j1,k,k+1} \sum_{i=0}^{n} w_{ji} x_i(t). \qquad (3)$$

Згідно з виразом (3), в один момент часу t активізуються лише дві сусідні функції належності: $g_{jk}(z_j(t))$ та $g_{j,k+1}(z_j(t))$, отже, тільки два набори ваг: u_{jk} та $u_{j,k+1}$ відповідно, потребують налаштування. Таким чином, загальна складність процесу навчання не залежить від кількості функцій m, а пропорційна значенню $n + 3$, де n визначається кількістю вхідних сигналів нейрона. Треба зазначити, що загальна кількість параметрів елементарного персептрона складає $n + 1$, а отже, відмінність у складності навчання між таким персептроном та нейроном з APWA незначна.

Таким чином, зв'язок між вагами u_{jk}, $u_{j,k+1}$ та $\nu_{j0,k,k+1}$,$\nu_{j1,k,k+1}$ сусідніх функцій визначається простими виразами:

$$\begin{cases} u_{jk} & = \nu_{j0,k,k+1} + c_{jk}\nu_{j1,k,k+1}, \\ u_{j,k+1} & = \nu_{j0,k,k+1} + c_{j,k+1}\nu_{j1,k,k+1}. \end{cases}$$

Розглянемо процес навчання нейрона з APWA на основі мінімізації традиційного локального квадратичного критерію:

$$J_j(t) = 0,5\ell_j^2(t) = 0,5\left(y_j(t) - \hat{y}_j(t)\right)^2 = 0,5\left(y_j(t) - \nu_{j0} - \nu_{j1} w_j^{\mathrm{T}} x(t)\right)^2 =$$

$$= 0,5\left(y_j(t) - \nu_{j0} - \nu_{j1} z_j(t)\right)^2 = 0,5\left(y_j(t) - u_j^{\mathrm{T}} g_j(t)\right)^2,$$

де $y_j(t)$ — еталонний сигнал, $\ell_j(t)$ — різниця між еталоном та сигналом на виході нейрона, а індекси $k, k+1$ значень $\nu_{j0,k,k+1}$, $\nu_{j1,k,k+1}$ опущені для стислості запису.

Ваги u_j нелінійного синапсу APWA в j-му нейроні можуть бути налаштовані за градієнтною процедурою:

$$u_{jk}(t) = u_{jk}(t-1) + \tilde{\zeta}(t)\ell_j(t)g_{jk}(z_j(t)),$$

або у векторній формі:

$$u_j(t) = u_j(t-1) + \tilde{\zeta}(t)\left(y_j(t) - u_j^{\mathrm{T}}(t-1)g_j(t)\right) g_j(t),$$

де $\tilde{\zeta}(t)$ — параметр швидкості навчання, $g_j(t)$ — вихід функції належності нейрона.

Обчислення можуть бути спрощені шляхом налаштування на кожному кроці t лише тих синаптичних ваг, котрі відповідають активним функціям належності:

$$\nu_{j0}(t) = \nu_{j0}(t-1) + \tilde{\zeta}(t)\left(y_j(t) - \nu_{j0}(t-1) - \nu_{j1}(t-1)z_j(t)\right) =$$

$$= \nu_{j0}(t-1) + \tilde{\zeta}(t)\ell_j(t),$$

$$\nu_{j1}(t) = \nu_{j1}(t-1) + \tilde{\zeta}(t)\left(y_j(t) - \nu_{j0}(t-1) - \nu_{j1}(t-1)z_j(t)\right) z_j(t) =$$

$$= \nu_{j1}(t-1) + \tilde{\zeta}(t)\ell_j(t)z_j(t).$$

Для налаштування синаптичних ваг w_j першого шару нейрона використовується стандартне дельта-правило [14]:

$$w_{ji}(t) = w_{ji}(t-1) + \zeta(t)\delta_j(t)x_i(t),$$

де $\delta_j(t) = \ell_j(t)\varphi_j'(z_j(t)) = \ell_j(t)\nu_{j1}(t)$, $\zeta_j(t)$ — параметр швидкості навчання для синаптичних ваг. У векторній формі маємо

$$w_j(t) = w_j(t-1) + \zeta(t)\ell_j(t)\nu_{j1}(t)x(t). \qquad (4)$$

З виразу (4) видно, що дельта-похибка лінійно залежить від похибки на виході

нейрона, що дозволяє спростити та прискорити процес навчання нейронів з APWA порівняно з елементарним персептроном з сигма-функцією.

2. Багатошаровий персептрон на основі нейронів з APWA

Розглянемо загалом процес навчання на прикладі тришарової архітектури штучної нейронної мережі. На рис. 2 ілюструється її загальна структура.

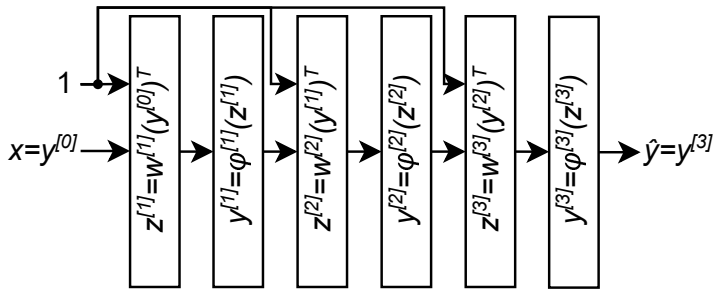

Рис. 2

Мережа включає в себе $n_0 = n$ входів, n_1 нейронів з APWA — у першому прихованому шарі, n_2 — у другому та $n_3 = h$ — у вихідному шарі. Кожен вхідний елемент представлений $(n + 1) \times 1$-вектором $x(t) = \big(1, x_1(t), \ldots, x_i(t), \ldots, x_n(t)\big)^{\mathrm{T}}$, а вихідний та еталонний сигнали — $h \times 1$-векторами $\hat{y}(t) = \big(\hat{y}_1(t), \hat{y}_2(t), \ldots, \hat{y}_j(t), \ldots, \hat{y}_h(t)\big)^{\mathrm{T}}$ та $y(t) = \big(y_1(t), y_2(t), \ldots, y_j(t), \ldots, y_h(t)\big)^{\mathrm{T}}$.

Як цільова функція використовується критерій

$$J(t) = \sum_{j=1}^{h} J_j(t) = 0,5 \sum_{j=1}^{h} \big(y_j(t) - \hat{y}_j(t)\big)^2 .$$

Оптимізація цільової функції відбувається шляхом налаштування синаптичних ваг $w_{ji}^{[l]}$ та $\nu_{jp}^{[l]}$, де $p = 0, 1$, $l = 1, 2, 3$ — індекс шару, h_j — число нейронів у шарі l, $j = 1, \ldots, h_l$ — індекс нейрона в шарі, $i = 0, 1, \ldots, n_l$ — індекс входу нейрона.

Процес навчання заснований на процедурі зворотного поширення помилки. На відміну від стандартної процедури для елементарного персептрона, налаштування нейрона з APWA включає налаштування параметрів активаційної функції. Процедура в загальному випадку може бути визначена як

$$\begin{cases} \Delta\nu_{jp}^{[l]}(t) = \nu_{jp}^{[l]}(t) - \nu_{jp}^{[l]}(t-1) = -\tilde{\zeta}^{[l]}(t)\dfrac{dJ(t)}{d\nu_{jp}^{[l]}}, \\ \Delta w_{jp}^{[l]}(t) = w_{ji}^{[l]}(t) - w_{ji}^{[l]}(t-1) = -\zeta^{[l]}(t)\dfrac{dJ(t)}{dw_{ji}^{[l]}}, \end{cases}$$

де основним кроком є обчислення часткових похідних $\dfrac{dJ(t)}{d\nu_{jp}^{[l]}}$, $\dfrac{dJ(t)}{dw_{ji}^{[l]}}$.

Найпростішою, з точки зору обчислень, є процедура оновлення для синаптичних ваг вихідного шару ($l = 3$):

$$\begin{cases} \Delta\nu_{jp}^{[3]}(t) = -\tilde{\zeta}^{[3]}(t)\dfrac{dJ(t)}{d\hat{y}_j(t)}\dfrac{d\hat{y}_j(t)}{d\nu_{jp}^{[3]}}, \\ \Delta w_{ji}^{[3]}(t) = -\zeta^{[3]}(t)\dfrac{dJ(t)}{dz_j^{[3]}(t)}\dfrac{dz_j^{[3]}(t)}{dw_{ji}^{[3]}}, \end{cases}$$

або у розширеному вигляді:

$$\begin{cases} \Delta\nu_{j0}^{[3]}(t) = \tilde{\zeta}^{[3]}(t)\ell_j(t), \\ \Delta\nu_{j1}^{[3]}(t) = \tilde{\zeta}^{[3]}(t)\ell_j(t)z_j^{[3]}(t), \\ \Delta w_{ji}^{[3]}(t) = \zeta^{[3]}(t)\delta_j^{[3]}(t)y_i^{[2]}(t) = \zeta^{[3]}(t)\ell_j(t)\nu_{j1}^{[3]}(t)y_i^{[2]}(t). \end{cases} \tag{5}$$

Аналогічно визначимо процедуру для другого прихованого шару ($l = 2$):

$$\begin{cases} \Delta \nu_{jp}^{[2]}(t) &= -\tilde{\zeta}^{[2]}(t)\frac{dJ(t)}{d\nu_{jp}^{[2]}} = -\tilde{\zeta}^{[2]}(t)\frac{dJ(t)}{dy_j^{[2]}(t)}\frac{dy_j^{[2]}(t)}{d\nu_{jp}^{[2]}}, \\ \Delta w_{ji}^{[2]}(t) &= -\zeta^{[2]}(t)\frac{dJ(t)}{dw_{ji}^{[2]}} = -\zeta^{[2]}(t)\frac{dJ(t)}{dz_j^{[2]}(t)}\frac{dz_j^{[2]}(t)}{dw_{ji}^{[2]}} = \\ &= -\zeta^{[2]}(t)\frac{dJ(t)}{dy_j^{[2]}(t)}\frac{dy_j^{[2]}(t)}{dz_j^{[2]}(t)}y_i^{[1]}(t) = \zeta^{[2]}(t)\delta_j^{[2]}(t)y_i^{[1]}(t). \end{cases} \tag{6}$$

Основним кроком тут є визначення часткової похідної з рівняння (6):

$$-\frac{dJ(t)}{dy_j^{[2]}(t)} = -\sum_{i=1}^{h_3}\left(\frac{dJ(t)}{dz_i^{[3]}(t)}\frac{dz_i^{[3]}(t)}{dy_j^{[2]}}\right) =$$

$$= \sum_{i=1}^{h_3}\left(-\frac{dJ(t)}{dz_i^{[3]}(t)}\right)\frac{d}{dy_j^{[2]}}\left(\sum_{s=1}^{h_2} w_{is}^{[1]}(t)y_s^{[2]}(t)\right) = \sum_{s=1}^{h_2}\delta_s^{[3]}(t)w_{sj}^{[3]}(t). \tag{7}$$

Звідси отримуємо наступне:

$$\delta_j^{[2]}(t) = \frac{d\varphi^{[2]}\left(z_j^{[2]}(t)\right)}{dz_j^{[2]}}\sum_{i=1}^{h_3}\delta_i^{[3]}(t)w_{ij}^{[3]}(t) = \nu_{j1}^{[2]}(t)\sum_{i=1}^{h_3}\delta_i^{[3]}(t)w_{ij}^{[3]}(t),$$

де $\nu_{j0}^{[2]}$ та $\nu_{j1}^{[2]}$ визначаються як

$$\nu_{j0}^{[2]}(t) = \nu_{j0}^{[2]}(t-1) + \tilde{\zeta}^{[2]}(t)\left(\sum_{i=1}^{h_3}\delta_i^{[3]}(t)w_{ij}^{[3]}(t)\right),$$

$$\nu_{j1}^{[2]}(t) = \nu_{j1}^{[2]}(t-1) + \tilde{\zeta}^{[2]}(t)\left(\sum_{i=1}^{h_3}\delta_i^{[3]}(t)w_{ij}^{[3]}(t)\right)z_j^{[2]}(t).$$

Налаштування першого прихованого шару ($l = 1$) виконується за відношенням

$$\begin{cases} \Delta \nu_{j0}^{[1]}(t) &= \tilde{\zeta}^{[1]}(t)\left(\sum_{i=1}^{h_2}\delta_i^{[2]}(t)w_{ij}^{[2]}(t)\right), \\ \Delta \nu_{j1}^{[1]}(t) &= \tilde{\zeta}^{[1]}(t)\left(\sum_{i=1}^{h_2}\delta_i^{[2]}(t)w_{ij}^{[2]}(t)\right)z_j^{[1]}(t), \\ \Delta w_{ji}^{[1]}(t) &= \zeta^{[1]}(t)\delta_j^{[1]}(t)x_i(t), \end{cases}$$

де $\delta_j^{[1]}(t) = \nu_{j1}^{[1]}(t)\sum_{i=1}^{h_2}\delta_i^{[2]}(t)w_{ij}^{[2]}(t)$.

У разі мережі з довільно великою кількістю шарів процес навчання шару з індексом l виконується за відношенням

$$\begin{cases} \Delta \nu_{j0}^{[l]}(t) &= \tilde{\zeta}^{[l]}(t)\left(\sum_{i=1}^{h_{l+1}}\delta_i^{[l+1]}(t)w_{ij}^{[l+1]}(t)\right), \\ \Delta \nu_{j1}^{[l]}(t) &= \tilde{\zeta}^{[l]}(t)\left(\sum_{i=1}^{h_{l+1}}\delta_i^{[l+1]}(t)w_{ij}^{[l+1]}(t)\right)z_j^{[l]}(t), \\ \Delta w_{ji}^{[l]}(t) &= \zeta^{[l]}(t)\delta_j^{[l]}(t)y_i^{[l-1]}(t), \\ \delta_j^{[l]}(t) &= \nu_{j1}^{[l]}(t)\sum_{i=1}^{h_{l+1}}(t)\delta_i^{[l+1]}w_{ji}^{[l+1]}(t). \end{cases} \tag{8}$$

Таким чином, відношення (8) є узагальненням виразів (5)–(7).

З точки зору процесу навчання, основною відмінністю запропонованої мережі порівняно з традиційним багатошаровим персептроном з довільною кількістю шарів є те, що дана мережа налаштовує активаційну функцію в процесі навчання відповідно до цільової функції та умов вирішуваної задачі.

3. Експериментальні дослідження

Порівнюємо продуктивність APWA з продуктивністю функції ReLU у складі повнозв'язного шару нейронів у двох моделях згорткових нейронних мереж (CNN):

LeNet-5 [15] та KerasNet [16]. Базові реалізації використовують нелінійну функцію ReLU для всіх згорткових та повнозв'язних шарів, окрім двох, останній з яких використовує Softmax для відображення класів на виході моделі. Похідні варіанти мереж застосовують APWA як активаційну функцію у передостанньому повнозв'язному шарі. Всі інші властивості мережі та процедури навчання залишаються спільними між усіма реалізаціями. Експеримент виконаний із застосуванням Python 3.10, бібліотеки PyTorch 2.0 [17] та комп'ютера з RTX A4000.

Продуктивність версій згорткових ШНМ оцінена на наборах даних Fashion-MNIST [18] та CIFAR-10 [19]. Розмірність та кількість даних, використаних в експерименті, співпадає з тими, що надані в роботах [18, 19], розбиття між кількістю вхідних даних для тренування до кількості даних для перевірки складає 5:1. Кожен з каналів вхідних зображень закодований у вигляді чисел з плаваючою точкою в діапазоні [0,0;+1,0]. Для значень класів зображень використовується пряме кодування (one hot encoding).

Дані, використані для тренування, доповнюються шляхом горизонтального віддзеркалення з вірогідністю 50 %, а також випадкового зсуву пікселів з максимальним зміщенням у 0,1 раз. Аугментація вимкнена під час оцінки продуктивності мережі на тестовому наборі даних (test set).

LeNet-5 має чотири шари:

• згортковий шар зі згорткою 5 x 5 та 20 вихідними каналами, вибірка за максимальним значенням 2 x 2, функція активації ReLU;

• згортковий шар зі згорткою 5 x 5 та 50 вихідними каналами, вибірка за максимальним значенням 2 x 2, функція активації ReLU;

• повнозв'язний шар з 500 вихідними ознаками, функція ReLU або APWA;

• повнозв'язний шар з 10 вихідними ознаками, активаційна функція SoftMax.

KerasNet складається з шести шарів:

• згортковий шар зі згорткою 3 x 3, 32 вихідними каналами та 1 x 1 відступом (pad), функція активації ReLU;

• згортковий шар зі згорткою 3 x 3, 32 вихідними каналами, без відступу, функція активації ReLU, вибірка (pooling) за максимальним значенням розміром 2 x 2 та dropout з вірогідністю 25 %;

• згортковий шар 3 x 3, 64 вихідних канали та 1x1 відступ, функція ReLU;

• згортковий шар 3 x 3, 64 вихідних канали, без відступу, функція ReLU, вибірка за максимальним значенням 2 x 2 та dropout ($p = 0,25$);

• повнозв'язний шар з 512 вихідними ознаками, активаційна функція ReLU або APWA, dropout на виході ($p = 0,5$);

• повнозв'язний шар з 10 вихідними ознаками, активаційна функція SoftMax.

Адаптивні функції активації в нейронах з APWA мають 14 налаштованих параметрів та 14 відповідних функцій належності. Ваги адаптивних функцій у різних нейронах повністю незалежні, отже, APWA в кожному нейроні (у кожному каналі) може отримати унікальну форму за результатами навчання.

Область визначення активаційної функції $[-1,0;+1,0]$ покривають 12 трикутних функцій належності, розподілених рівномірно. Значення за межами області визначення покриваються двома додатковими функціями типу «рампа» (ramp function):

$$g_{j0}(z_j) = \begin{cases} 1, & z_j \in (-\infty, c_{j0}), \\ \frac{c_{j1}-z_j}{c_{j1}-c_{j0}}, & z_j \in \left[c_{j0}, c_{j1}\right], \\ 0, & z_j \in (c_{j1}, +\infty), \end{cases}$$

$$g_{jm}(z_j) = \begin{cases} 0, & z_j \in (-\infty, c_{j0}), \\ \frac{z_j - c_{j,m-1}}{c_{jm} - c_{j,m-1}}, & z_j \in \left[c_{j,m-1}, c_{jm}\right], \\ 1, & z_j \in (c_{jm}, +\infty). \end{cases}$$

В експерименті порівнюється три способи ініціалізації початкових значень параметрів u_k в нейронах APWA:

• Ramp — у порядку рівномірного зростання від 0,0 до +1,0;

• Const1 — константою +1,0;

• Random — рівномірним випадковим розподілом у діапазоні $[-1, 0; +1, 0]$.

Решта параметрів, такі як синаптичні ваги w_j, ініціалізується відповідними процедурами за замовчуванням з PyTorch.

Навчання всіх варіантів мереж проводиться протягом 100 епох з розміром мінівибірки 64 елементи та оптимізатором RMSprop. Початкова швидкість навчання: $\zeta(0) = 1 \cdot 10^{-4}$, після обробки кожної мінівибірки з набору даних для навчання швидкість зменшується на $1 \cdot 10^{-6}$.

Для простоти приймаємо $\zeta(t) = \tilde{\zeta}(t)$, тобто значення швидкості навчання на кожному кроці t для всіх параметрів моделі співпадають.

4. Аналіз експериментальних даних

Реєструємо значення похибки на навчальному наборі даних та точність розпізнавання на тестовому наборі для кожного з варіантів мережі. Навчені моделі зберігаються для подальшого аналізу форми активаційних функцій.

У таблиці наведені результати навчання на наборах даних Fashion-MNIST та CIFAR-10 (найкраща отримана точність класифікації та епоха, за якої був отриманий цей показник точності) для кожного з оцінених варіантів мережі, наборів даних та стартових форм функцій.

Таблиця

		Fashion-MNIST		CIFAR-10	
Мережа	Активаційна функція	Точність, %	Епоха	Точність, %	Епоха
LeNet-5	ReLU	91,48	98	76,11	95
LeNet-5	APWA Ramp	91,30	92	76,90	98
LeNet-5	APWA Random	92,29	76	77,09	89
LeNet-5	APWA Const1	92,13	94	76,25	96
KerasNet	ReLU	91,21	75	79,43	99
KerasNet	APWA Ramp	92,24	98	79,80	100
KerasNet	APWA Random	93,19	94	82,32	93
KerasNet	APWA Const1	92,20	92	79,80	97

Загалом варіанти мережі з нейронами на основі APWA показують кращі результати порівняно з базовими реалізаціями, що використовують ReLU, як на наборі даних CIFAR-10, так і Fashion-MNIST. Серед всіх варіантів ініціалізації APWA найкраще себе зарекомендував варіант з випадковою ініціалізацією параметрів.

На рис. 3, 4 (процес навчання мережі архітектури KerasNet на наборах Fashion-MNIST і CIFAR-10 відповідно) ілюструють залежність показників точності на тестовому наборі (test accuracy) та похибки навчання (training loss) від епохи для KerasNet.

Варто зазначити, що реалізація KerasNet з параметрами функції APWA, ініціалізованими випадково, навчається швидше та показує меншу тенденцію до перенавчання (overfitting) порівняно з іншими активаційними функціями у всіх випадках, окрім LeNet-5 на наборах Fashion-MNIST та CIFAR-10. Еталонний варіант мережі KerasNet з ReLU так само показує високу схильність до перенавчання на наборі Fashion-MNIST.

З даних рис. 3 та 4 видно, що константна ініціалізація затримує процес навчання та підвищує похибку на початкових етапах, оскільки APWA-функція з такими

вагами не змінює значення на виході в залежності від входу, а отже, блокує пряме проходження інформації, поки градієнтна процедура не адаптує ваги.

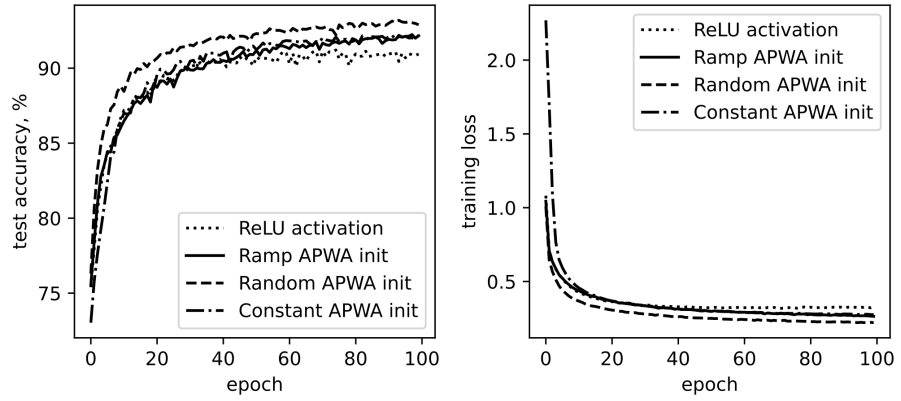

Рис. 3

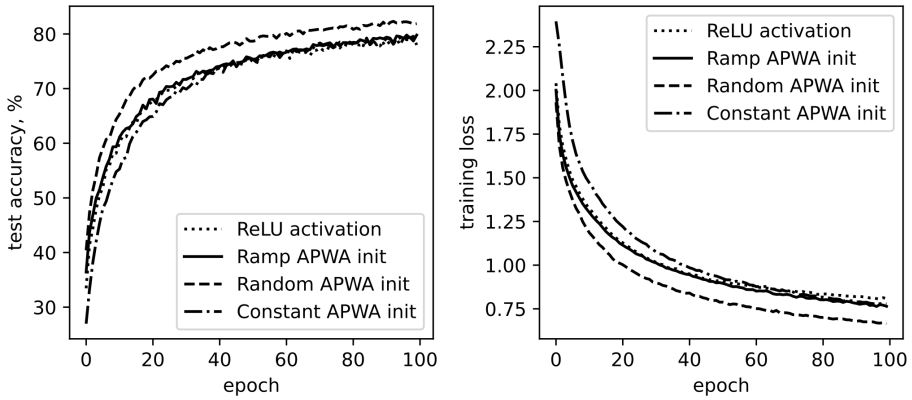

Рис. 4

На рис. 5 зображено форму функції APWA, ініціалізованої константними параметрами, за результатами навчання. Цікавою особливістю є синусоїдо-подібна форма функції, особливо виражена в мережі KerasNet на наборі CIFAR-10.

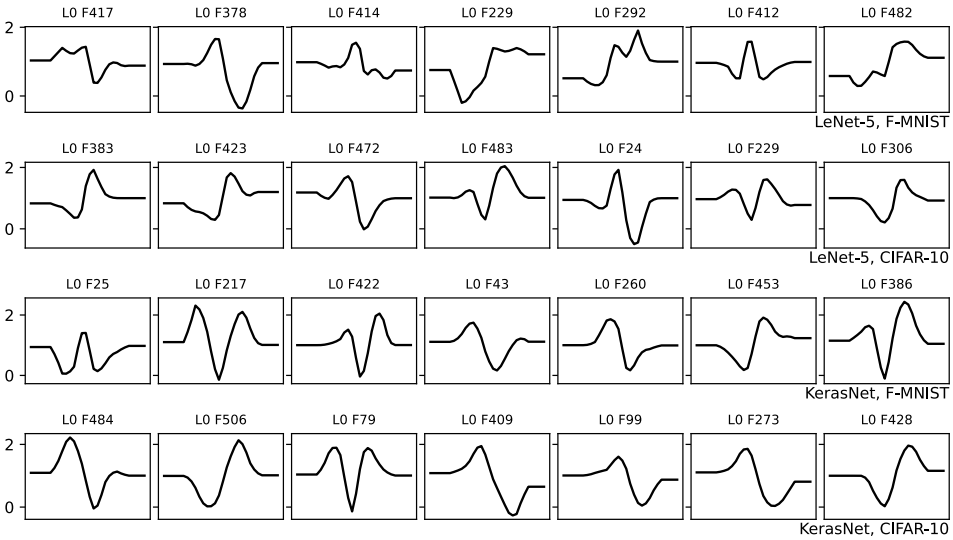

Рис. 5

На рис. 6 представлено форму функції APWA, ініціалізованої випадково, за результатами навчання, що показала себе найкращою в процесі оцінки точності. Варто зазначити сильну нелінійність отриманих функцій.

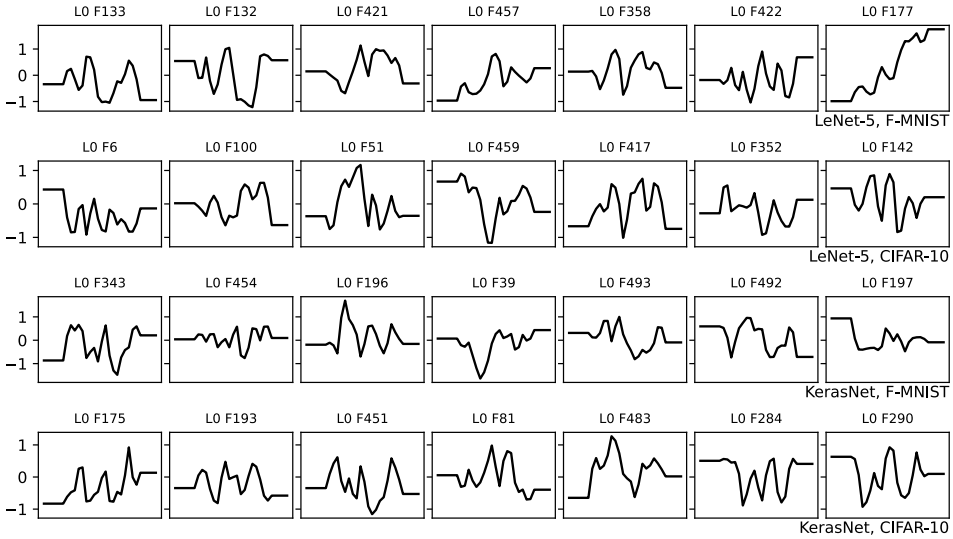

Рис. 6

На рис. 7 показано форму функції APWA, ініціалізованої як Ramp-функція, за результатами навчання. У цьому разі функція зберігає свою загальну форму, але з набором даних CIFAR-10 деформується в районі $x \approx 0$.

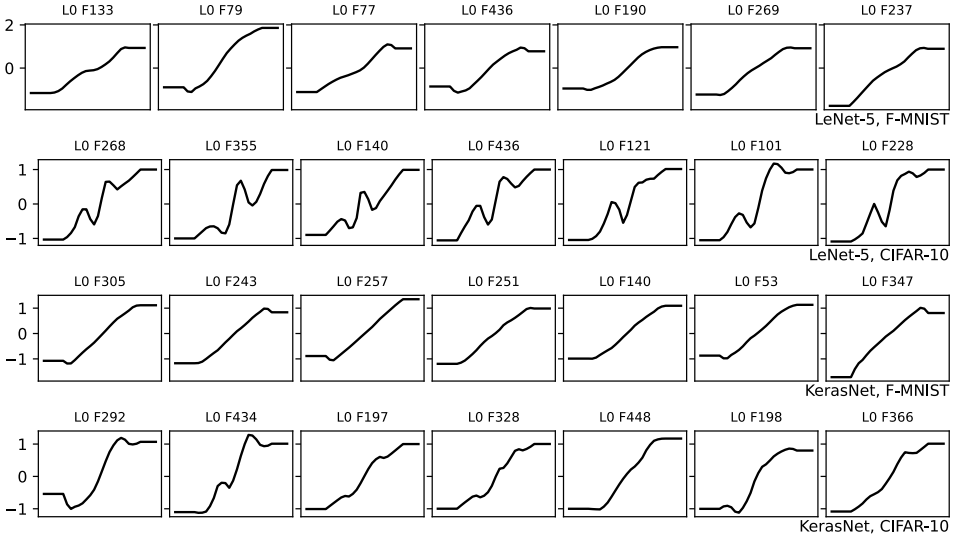

Рис. 7

Висновок

Представлено APWA та нейрон на її основі як альтернатива стандартним кусково-заданим активаційним функціям, таким як ReLU, що широко застосовуються в глибоких нейронних мережах. Основною відмінністю мереж з APWA, порівняно з елементарним персептроном Ф. Розенблата, є використання нелінійних синапсів на основі нечіткого перетворення замість фіксованих активаційних функцій.

Синапси APWA реалізують нечітке перетворення в його адаптивній формі, а також дозволяють виконувати апроксимацію існуючих активаційних функцій на інтервалі з довільним рівнем точності, що визначається кількістю функцій належності.

Форма активаційної функції APWA налаштовується згідно з поточною задачею окремо для кожного з нейронів мережі, що надає їй поглиблені апроксимаційні можливості, а також дозволяє отримати кращі результати (точності тощо) в процесі навчання, що експериментально підтверджено в даній роботі.

Y. Bodyanskiy, S. Kostiuk

NEURON BASED ON AN ADAPTIVE FUZZY TRANSFORM FOR MODERN ARTIFICIAL NEURAL NETWORK MODELS

Yevgeniy Bodyanskiy

Kharkiv National University of Radio Electronics,
yevgeniy.bodyanskiy@nure.ua

Serhii Kostiuk

Kharkiv National University of Radio Electronics,
serhii.kostiuk@nure.ua

As the industry deals with growing datasets and more complex data processing challenges, researchers and enterprise specialists trade artificial neural network model complexity and speed for increased approximation capabilities. An increase in the model complexity, particularly in its number of layers and depth, leads to undesired computational effects like vanishing and exploding gradients. The production models often employ piece-wise linear activation functions, similar to ReLU, to avoid computational difficulties and improve the learning speed. While the piece-wise linear activation functions have proven their effectiveness in production models, namely in convolutional neural networks (CNN), such functions are usually constrained in their form, limiting the model's ability to optimize and adapt to the current task. We propose an Adaptive Piece-Wise Activation (APWA) function as an adaptive replacement for the fixed piece-wise linear activation functions. The core of the APWA function is an adaptive fuzzy transform of the input signal, implemented by a set of membership functions with adaptive output gains. Like the piece-wise linear units, APWA does not suffer from the effects of exploding and vanishing gradients and is relatively simple to compute, reducing the learning and inference time for networks with APWA-based neurons. We demonstrate the effectiveness of APWA-based neurons on two different image classification sets and two model architectures of distinct complexities. During the learning, the models with APWA adapt their activation function form, providing improved classification accuracy compared to the baseline non-adaptive variants.

Keywords: neural network model, adaptive fuzzy transform, activation function.

ПОСИЛАННЯ

1. Aggarwal C.C. Neural networks and deep learning. Springer International Publishing, 2023. P. 1–27. DOI: https://doi.org/10.1007/978-3-031-29642-0.
2. Rosenblatt F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review.* 1958. Vol. 65, N 6. P. 386–408. DOI: https://doi.org/10.1037/h0042519.
3. Cybenko G. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems.* 1989. Vol. 2, N 4. P. 303–314. DOI: https://doi.org/10.1007/BF02551274.

4. Hornik K. Approximation capabilities of multilayer feedforward networks. *Neural Networks.* 1991. Vol. 4, N 2. P. 251–257. DOI: https://doi.org/10.1016/0893-6080(91)90009-t.
5. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. URL: http://www.deeplearningbook.org.
6. Language Models are Few-Shot Learners / Brown T., Mann B., Ryder N., Subbiah M., Kaplan J.D., Dhariwal P., Neelakantan A., Shyam P., Sastry G., Askell A., Agarwal S., Herbert-Voss A., Krueger G., Henighan T., Child R., Ramesh A., Ziegler D., Wu J., Winter C., Hesse C., Chen M., Sigler E., Litwin M., Gray S., Chess B., Clark J., Berner C., McCandlish S., Radford A., Sutskever I., and Amodei D. *Advances in Neural Information Processing Systems* / ed. by Larochelle H., Ranzato M., Hadsell R. et al. Curran Associates, Inc. 2020. Vol. 33. P. 1877–1901. URL: https://papers.nips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.
7. CoAtNet: marrying convolution and attention for all data sizes / Dai Z., Liu H., Le Q.V., and Tan M. *Advances in Neural Information Processing Systems* / ed. by Ranzato M., Beygelzimer A., Dauphin Y. et al. Curran Associates, Inc. 2021. Vol. 34. P. 3965–3977. URL: https://papers.nips.cc/paper/2021/hash/20568692db622456cc42a2e853ca21f8-Abstract.html.
8. Agarap A.F. Deep learning using rectified linear Units (ReLU). *CoRR.* 2018. arXiv : 1803.08375.
9. Bodyanskiy Y., Chala O. Adaptive double neo-fuzzy neuron and its combined learning. *Системи управління, навігації та зв'язку. Збірник наукових праць.* 2023. Vol. 3, N 73. P. 70–74. DOI: https://doi.org/https://doi.org/10.26906/sunz.2023.3.070.
10. Picton P. ADALINE. *Introduction to Neural Networks.* Macmillan Education UK, 1994. P. 13–24. DOI: https://doi.org/10.1007/978-1-349-13530-1_2.
11. A neo-fuzzy neuron and its application to system identification and prediction of the system behavior / Yamakawa T., Uchino E., Miki T., and Kusanagi H. *Proc. 2nd Int. Conf. on Fuzzy Logic and Neural Networks.* 1992. P. 477–483.
12. Bodyanskiy Y.V., Tyshchenko O.K., Kopaliani D.S. Adaptive learning of an evolving cascade neo-fuzzy system in data stream mining tasks. *Evolving Systems.* 2016. Vol. 7, N 2. P. 107–116. DOI: https://doi.org/10.1007/s12530-016-9149-5.
13. Perfilieva I. F-Transform. *Springer Handbook of Computational Intelligence* / ed. by Kacprzyk J., Pedrycz W. Berlin Heidelberg : Springer, 2015. P. 113–130. DOI: https://doi.org/10.1007/978-3-662-43505-2_7.
14. Cichocki A., Unbehauen R. Neural networks for optimization and signal processing. 1st ed. USA : John Wiley & Sons, Inc., 1993. ISBN: 0471930105.
15. Gradient-based learning applied to document recognition / Lecun Y., Bottou L., Bengio Y., and Haffner P. *Proceedings of the IEEE.* 1998. Vol. 86, N 11. P. 2278–2324. DOI: https://doi.org/10.1109/5.726791.
16. Chollet F. et al. Train a simple deep CNN on the CIFAR10 small images dataset. Keras. GitHub, 2015. URL: https://github.com/keras-team/keras/blob/1.2.2/examples/cifar10_cnn.py.
17. PyTorch: an imperative style, high-performance deep learning library / Paszke A., Gross S., Massa F., Lerer A., Bradbury J., Chanan G., Killeen T., Lin Z., Gimelshein N., Antiga L., Desmaison A., Kopf A., Yang E., DeVito Z., Raison M., Tejani A., Chilamkurthy S., Steiner B., Fang L., Bai J., and Chintala S. *Advances in Neural Information Processing Systems 32* / ed. by Wallach H., Larochelle H., Beygelzimer A. et al. Curran Associates, Inc., 2019. P. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.
18. Xiao H., Rasul K., Vollgraf R. Fashion-MNIST: a novel smage dataset for benchmarking machine learning algorithms. *CoRR.* 2017. Vol. abs/1708.07747. arXiv : 1708.07747.
19. Learning multiple layers of features from tiny images : Rep. : 0 / University of Toronto; executor: Krizhevsky A., Hinton G. Toronto, Ontario : 2009. URL: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

Отримано 28.10.2023