Karafan

Quarterly Scientific Journal of National University of Skills

Autumn 2024, Vol. 21, No. 3, p. 57-83

Journal Homepage: https://karafan.nus.ac.ir/?lang=en

doi:10.48301/KSSA.2024.427012.2773

Presenting an "Adaption Ahead" Optimization Algorithm for Training Models Used in Deep Learning

Farnaz Hoseini[1*], Hamed Sepehrzadeh[2], Masume Kheyri[3]

[1,2]Assistant Professor, Department of Computer Engineering, National University of Skills (NUS), Tehran, Iran.

[3]The Coach, Department of Computer Engineering, National University of Skills (NUS), Tehran, Iran.

ARTICLE INFO

Article Type:
Original Research

Received: 12.05.2023
Revised: 01.17.2024
Accepted: 04.05.2024

Keyword:
Deep Learning
Optimization
Optimization Algorithms
Deep Models
Adaption Ahead Algorithm

***Corresponding Author:**
Farnaz Hoseini
Email: *f-hoseini@tvu.ac.ir*

ABSTRACT

Deep learning is a subset of machine learning that is widely used in the field of artificial intelligence such as natural language processing and machine vision. As a subset of machine vision, image segmentation is one of the most common steps in digital image processing, which divides a digital image into different segments. In this research, a new method based on deep learning for image segmentation was presented. The "Adaption Ahead" algorithm was introduced and used as a new optimization algorithm to optimize the proposed model. In previous optimization algorithms, the most important factor in reducing accuracy was extracting low-level features of images and not reducing the semantic distance between human perception and features. In this study, hierarchical and deep feature extraction from images was carried out with the help of deep learning. The "Adaption Ahead" optimization algorithm, in which a deep model based on a convolutional neural network is used, extracted higher-level features and achieved optimal accuracy. By using the Nestrov technique in calculating the gradient by the proposed algorithm, the best result, i.e. 91.1 accuracy, was obtained for the Dice similarity measure. Another advantage of this algorithm over other methods was using uncomplicated calculations. The comparison of the proposed optimization algorithm with other commonly used methods demonstrated the improvement in the performance of this network on relatively large data sets. Furthermore, the more accurate performance of this network, as a result of its hierarchical and deep extraction, was compared to other methods.

E-ISSN: 2538-4430
ISSN: 2382-9796

EXTENDED ABSTRACT

Introduction

The basic concept of deep learning originates from artificial neural networks, with the difference that in this concept there are greater discussions such as deep neural network, Convolutional Neural Network (CNN) and Deep Belief Network (DBN). The present research findings illustrated that algorithms with an effective learning rate are generally more powerful than other algorithms. The most common optimization algorithms used in the process of training deep models are the SGD algorithm (Stochastic Gradient Descent), Momentum algorithm, Nestrove algorithm, AdaGrad algorithm, RMSProp algorithm and Adam algorithm. The choice of these algorithms mostly depends on familiarity with the application of each algorithm and how to arrange their meta parameters. Improving optimization algorithms is not always the best way to improve the optimization process. In deep models, the model is designed to make optimization easier.

Methodology

The architecture of the deep model presented in this study was a model based on a convolutional neural network (CNN) which is presented to check the performance of the proposed algorithm. In order to find optimal values for model parameters, an optimization algorithm called "Adaptation Ahead" is proposed. The name "Adaptation Ahead" was selected for the proposed optimization algorithm because by defining an objective function for the algorithm, it attempts to calculate the values of the adjustable parameters in such a way that the desired objective function according to the experimental data set is minimized.

Table 1. The proposed optimization algorithm "Adaptation Ahead".

1	*Input Dataset (N examples)*
2	$\epsilon \leftarrow 10^{-8}, \lambda_v \leftarrow 0.999, \lambda_m \leftarrow 0.9$
3	$\alpha_0 = 0.01$
4	$p = or_{sw1}\{1, 2, \infty\}$
5	$\theta_0 \leftarrow N(0, \sqrt{2/N})$
6	$t, m_0, \upsilon_0 \leftarrow 0$
7	***while*** *(Reduce Error in Validation Data)*
8	$t = t + 1$
9	$g_t \leftarrow \nabla_\theta L(\theta_{t-1} - \alpha_t \lambda_m m_{t-1} or_{sw2}\{0, 1\})$
10	$m_t \leftarrow \lambda_m m_{t-1} + (1 - \lambda_m) g_t$
11	$v_t \leftarrow \lambda_v v_{t-1} + (1 - \lambda_v) \lvert g_t \rvert^p$
12	$\Delta_t \leftarrow \alpha_t m_t or_{sw3} \left\{ \frac{1}{\sqrt[p]{v_t} + \epsilon}, 1 \right\}$
13	$\theta_t = \theta_{t-1} - \Delta_t$
14	***End while***

The details and steps related to the proposed optimization algorithm "forward adaptation" are shown in Table 1. ϵ is added for numerical stability and prevents the denominator of the fraction from becoming zero in step (line) 12. λ_m is the first-moment calculation constant, which is set to 0.9 to create a relatively small sliding window for

calculating the average moment. λ_v is the second moment calculation constant, which is set to 0.999 to create a relatively long sliding window for calculating the average moment. Then, the initial value was given to the parameters or the main variables of the model. The learning rate in the presented algorithm was initially set to 0.01 and then it was changed iteratively. In other words, in the iterations of the algorithm, the learning rate value was multiplied by 0.1 by observing the stabilization of the error rate in the validation data. The soft number is the meta parameter of the algorithm and is set in line 4. The values considered for this meta parameter were 1, 2, and infinity, which correspond to soft 1, soft Euclidean, and soft maximum, and are selected by switch 1 (sw1). In the findings of the research (fourth part), the behavior of the algorithm was evaluated for each of these values. In step 5, the weights of the model were set with Gaussian random values with zero mean and 2/N variance. In this relation, N is the number of inputs of the corresponding neuron. The algorithm continues as long as the error rate in the validation data is decreasing. In line 9, switch 2 (sw2) determines whether the gradient should be calculated in the normal or Nestrov way. In line 10, the first moment is calculated, which actually calculates the gradient. In line 11, the moment of order p is calculated. In line 12, switch 3 (sw3) determines whether the learning rate will operate adaptively using the moment calculated in line 12 or operate in the usual way based on Nestrov. In this way, by setting the built-in switches, you can examine eight different combinations. In the table below, it can be seen that by deactivating the adaptive learning rate, the soft type becomes unaffected.

Results and discussion

The experimental process was carried out using a NVIDIA GeForce series GeForce GTX 1080 Ti. To evaluate the algorithm and the proposed model, using the BRATS dataset, the segmentation of the four-dimensional MR images was performed and the Dice similarity metric was used to compare the performance. In Table 2, corresponding to the selection of each switch in the proposed optimization algorithm of the present paper, the results obtained from the training of the DCNN model are presented.

Table 2. The results of the accuracy obtained from the proposed optimization algorithm for selecting different switches.

Exp. No.	Switches States			$Accuranc = \frac{\lvert P \wedge T \rvert + \lvert (A-P) \wedge (A-T) \rvert}{\lvert A \rvert}$
	SW1(Norm)	SW2(Nestrov)	SW3(Adaptive)	
(1)	-	Yes	No	85.1
(2)	-	No	No	83.5
(3)	1	Yes	Yes	87.3
(4)	1	No	Yes	86.1
(5)	2	Yes	Yes	**91.1**
(6)	2	No	Yes	88.2
(7)	∞	Yes	Yes	86.4
(8)	∞	No	Yes	85.5

Conclusion

In this study, a deep model was proposed. To train the proposed model, the cost function should reach its minimum despite being non-linear, and for this, the "Adaptation Ahead" algorithm was used as an optimization algorithm. The basic random gradient descent

algorithm moves towards local minima in proportion to the negative value of the gradient, but it may be slow in areas with low curvature. In this study, hierarchical and deep feature extraction from images was carried out with the help of deep learning. The "forward matching" optimization algorithm, in which a deep model based on a convolutional neural network is used, extracted higher-level features and achieved optimal accuracy. By using the Nestrov technique in calculating the gradient by the proposed algorithm, the best result, i.e. 91.1 accuracy, was obtained for the Dice similarity measure.

کارافن
فصلنامه علمی دانشگاه ملی مهارت
پاییز ۱۴۰۳، دوره ۲۱، شماره ۳، ۸۳-۵۷
آدرس نشریه: https://karafan.nus.ac.ir/
doi:10.48301/KSSA.2024.427012.2773

ارائه الگوریتم بهینه‌سازیِ «انطباقِ پیشِ رو» برای آموزش مدل‌های مورد استفاده در یادگیری عمیق

فرناز حسینی[1*]، حامد سپهرزاده[2]، معصومه خیری[3]

۱و۲- استادیار، گروه مهندسی کامپیوتر، دانشگاه ملی مهارت، تهران، ایران.
۳- مربی، گروه مهندسی کامپیوتر، دانشگاه ملی مهارت، تهران، ایران.

اطلاعات مقاله

نوع مقاله: مقاله پژوهشی

دریافت مقاله: ۱۴۰۲/۰۹/۱۴
بازنگری مقاله: ۱۴۰۲/۱۰/۲۷
پذیرش مقاله: ۱۴۰۳/۰۱/۱۷

کلید واژگان:
یادگیری عمیق
بهینه‌سازی
الگوریتم‌های بهینه‌سازی
مدل‌های عمیق
الگوریتم انطباق پیش‌رو

***نویسنده مسئول:** فرناز حسینی
پست الکترونیکی:
f-hoseini@tvu.ac.ir

چکیده

یادگیری عمیق یکی از زیرمجموعه‌های یادگیری ماشینی است که به طور وسیع در زمینه هوش مصنوعی، مانند پردازش زبان طبیعی و بینایی ماشین بکار می‌رود. قطعه‌بندی تصویر به عنوان زیر مجموعه‌ای از بینایی ماشین، یکی از رایج ترین مراحل پردازش تصویر دیجیتال است، که یک تصویر دیجیتال را به قطعات مختلف تقسیم می‌کند. در این پژوهش یک روش نوین مبتنی بر یادگیری عمیق برای قطعه‌بندی تصاویر ارائه شده است. برای بهینه‌سازی مدل پیشنهادی الگوریتم "انطباقِ پیشِ رو"به عنوان یک الگوریتم بهینه‌سازی جدید معرفی و مورد استفاده قرار گرفت. در الگوریتم‌های بهینه‌سازی پیشین مهم‌ترین عامل کاهش دقت، استخراج ویژگی‌های سطح پایین تصاویر و عدم کاهش فاصله معنایی میان ادراک انسان و این ویژگی‌هاست. در این مطالعه با کمک یادگیری عمیق، استخراج سلسله مراتبی و عمیق ویژگی از تصاویر انجام شد. الگوریتم بهینه‌سازی "انطباقِ پیشِ رو"، که در آن از یک مدل عمیق مبتنی بر شبکه عصبی همگشتی استفاده شده، ویژگی‌های سطح بالاتری را استخراج کرده و به دقت مطلوبی دست یافته است. با بکارگیری تکنیک Nestrov در محاسبه گرادیان توسط الگوریتم پیشنهادی بهترین نتیجه یعنی دقت ۹۱/۱ برای معیار شباهت دایس بدست آمد. برتری دیگر این الگوریتم نسبت به سایر روش‌ها، استفاده از محاسبات غیر پیچیده است. مقایسه الگوریتم بهینه‌سازی پیشنهادی با سایر روش‌های معمول بکار رفته، نشان دهنده بهبود عملکرد این شبکه بر روی مجموعه داده‌های نسبتا بزرگ است. همچنین عملکرد دقیق‌ترِ این شبکه، در نتیجه‌ی استخراج سلسله مراتبی و عمیق آن، نسبت به روش‌های دیگر است.

شاپای الکترونیکی: ۲۵۳۸-۴۴۳۰
شاپای چاپی: ۲۳۸۲-۹۷۹۶

مقدمه

مفهوم اولیه یادگیری عمیق به شبکه‌های عصبی مصنوعی باز می‌گردد، با این تفاوت که در این دیدگاه بیشتر بحث‌هایی چون شبکه عصبی عمیق، شبکه عصبی همگشتی (CNN[1]) [۱] و شبکه باور عمیق (DBN[2]) [۲] در میان است. یادگیری عمیق (DL[3]) یا همان یادگیری ژرف یکی از مباحث جدید در هوش مصنوعی و یادگیری ماشین است [۳]. یادگیری عمیق زیر شاخه‌ای از یادگیری ماشین و بر مبنای مجموعه‌ای از الگوریتم‌هاست که مفاهیم انتزاعی سطح بالا در دادگان را مدل می‌کند. این فرآیند با استفاده از یک گراف عمیق مدل می‌شود که خود دارای چندین لایه پردازشی متشکل از چندین لایه تبدیلات خطی و غیر خطی است. یادگیری عمیق در واقع همان یادگیری به وسیله شبکه‌های عصبی مصنوعی است که دارای لایه‌های پنهان[4] زیادی می‌باشد؛ هر چند اگر در لایه‌های یک شبکه عصبی عمیق جلوتر برویم، به مدل‌های پیچیده‌تر و کامل‌تری می‌رسیم [۴]. دلایل استقبال گسترده از یادگیری عمیق در سال‌های اخیر را می‌توان از چند دیدگاه مورد توجه قرار داد. در یادگیری ماشین و شناسایی الگو، ویژگی یک خاصیت قابل اندازه‌گیری از یک پدیده‌ی مشاهده شده است. انتخاب ویژگی‌های مستقل، جداکننده و حاوی اطلاعات مفید[5]، نقشی موثر در الگوریتم‌های رده بند و رگرسیون دارند [۵]. ساختن دستی ویژگی‌ها یا همان مهندسی ویژگی‌ها، کار وقت‌گیری بوده و اغلب دارای مبالغه است. بعلاوه این عمل برای هر نوع داده (تصویر، صوت، متن و بانک اطلاعاتی)، کار، حوزه و زبان باید دوباره تکرار شود. در یادگیری ماشین چنانچه ویژگی‌ها بطور خودکار یاد گرفته شوند، فرآیند یادگیری در حوزه‌های مختلف به راحتی خودکار می‌شود. عموما داده‌های خام، دارای ابعاد بالایی هستند و فضای داده‌ها معمولابا داده‌های خام پر می‌شوند. به این ترتیب مدل یادگیری بسرعت دچار بیش برازش[6] می‌شود [۶]. عموما برای جلوگیری از این پدیده، از مهندسی ویژگی‌ها یا ساده‌سازی مدل یادگیری، استفاده می‌شود. مدل‌های یادگیری عمیق بخوبی از عهده داده‌هایی با بعد بالا برمی‌آیند [۷]. علاوه بر موارد فوق، عواملی همچون استفاده از مجموعه داده‌های بزرگ، کامپیوترهای موازی و تنظیم و بهینه‌سازی دقیق منجر به نتایج خوب در یادگیری عمیق شده است. در این مدل‌ها بخاطر عدم وجود مرحله مهندسی ویژگی و داشتن پارامترهای یادگرفتنی فراوان، داده‌های بیشتری مورد نیاز است. این امر با داشتن منبع عظیم اطلاعات موجود در وب براحتی میسر می‌شود. از طرفی در یادگیریِ عمیق محاسبات ماتریسی فراوان مورد نیاز در شبکه‌های عصبی، براحتی توسط کامپیوترهای چند هسته‌ای و تکنولوژی GPU[7] تسریع شده است [۸]. در فرآیند استخراج ویژگی توسط مدل‌های یادیگیری عمیق، معمولا چند نوع هسته مختلف بر روی لایه ورودی اعمال می‌شوند. نکته مهم این است که این هسته‌ها به صورت پراکنده اعمال و پارامترها به اشتراک گذاشته می‌شوند، بنابراین بار محاسباتی در مقایسه با شبکه‌های عصبی معمولی به طور قابل توجهی کاهش می‌یابد. الگوریتم‌های یادگیری عمیق از نقطه نظرهای مختلف نیازمند بهینه‌سازی هستند. برای مثال عموماً انجام استنتاج مبتنی بر مدل، مستلزم حل کردن یک مساله بهینه‌سازی است. در یادگیری عمیق، مهمترین و مفصل‌ترین مساله‌ای که از طریق بهینه‌سازی حل می‌شود، آموزش شبکه‌های عصبی و مدل‌های عمیق است [۹]. آموزش یک مدلِ عمیق ممکن است چند ماه، زمانِ پردازشِ صدها کامپیوتر را به خود اختصاص دهد. در فرآیند آموزش مدل‌های عمیق برای جلوگیری از تطبیق بیش از حد، داده‌های آموزشی زیادی مورد نیاز است. بدین منظور با استفاده از مجموعه داده‌های یک دامنه‌ی مرتبط، آموزش اولیه انجام شده و سپس با استفاده از داده‌های حوزه مورد نظر، تنظیم دقیق انجام می‌شود. اساساً در فرآیند یادگیری، به دلیل

[1] Convolutional Neural Network
[2] Deep Belief Networks
[3] Deep Learning
[4] Hidden Layers
[5] Informative
[6] Over-Fitting
[7] Graphics Processing Unit

تعداد زیاد پارامترها و داده‌های آموزشی، از تکنیک‌های پردازش موازی برای مقابله با پیچیدگی‌های محاسباتی و مکانی بالا استفاده می‌شود [۱۰].

در مطالعات اخیر به منظور آموزش مدل‌های عمیق از الگوریتم‌های پایه‌ی مختلفی برای حل مسئله بهینه‌سازی استفاده شده است. متأسفانه، هیچ توافق کلی در مورد مناسب‌ترین الگوریتم بهینه‌سازی وجود ندارد. در یک مطالعه قابل توجه [۷]، مقایسه‌ای بین تعداد زیادی از الگوریتم‌های بهینه‌سازی در مقابل وظایف مختلف انجام شده است. نتایج این مطالعه نشان می‌دهد که الگوریتم‌هایی که نرخ یادگیری مؤثری دارند، عموماً قوی‌تر از سایر الگوریتم‌ها هستند. متداول‌ترین الگوریتم‌های بهینه‌سازی مورد استفاده در فرآیند آموزش مدل‌های عمیق عبارتند از: الگوریتم SGD[1] (گرادیان نزولی تصادفی) [۱۱]، الگوریتم Momentum [۱۲]، الگوریتم Nestrove [۱۳]، الگوریتم AdaGrad [۱۴]، الگوریتم RMSProp [۱۵] و الگوریتم Adam [۱۶]. انتخاب این الگوریتم‌ها بیشتر به آشنایی با کاربرد هر الگوریتم و نحوه چیدمان فراپارامترهای آنها بستگی دارد. بهبود الگوریتم‌های بهینه‌سازی همیشه بهترین راه برای بهبود فرآیند بهینه‌سازی نیست. در مدل‌های عمیق، مدل به گونه‌ای طراحی می‌شود که بهینه‌سازی آسان‌تر انجام شود. به عبارت دیگر با انتخاب یک خانواده مدلِ مناسب می‌توان شرایط بهینه‌سازی ساده را با استفاده از الگوریتم دهه ۸۰ مانند SGD و ترکیبی از الگوریتم‌های جدید فراهم کرد. در این مطالعه ابتدا الگوریتم‌های مختلف بکار گرفته شده برای آموزش مدل‌های عمیق مورد بررسی قرار می‌گیرند، سپس با در نظر گرفتن نقاط قوت و ضعفِ این الگوریتم‌ها، یک الگوریتم بهینه‌سازی با عنوان "انطباقِ پیشِ رو" ارائه می‌گردد. در معرفیِ تکنیک‌های مختلف، یک معیار کارایی روی کل مجموعه داده‌های آموزشی تعریف می‌شود که به همراه عبارت تنظیم‌سازی، تابع هدف $J(\Theta)$ را تشکیل می‌دهد. در فرآیند بهینه‌سازی، پارامتر Θ (معیار کارایی) بگونه‌ای در نظر گرفته می‌شود که کمترین مقدار را برای این تابع هدف فراهم نماید. در الگوریتم پیشنهادی یک تابع هزینه ویژه نیز پیشنهاد شده که سعی دارد با استفاده از روش‌های بهینه‌سازی مبتنی بر گرادیان تطبیقی با سوئیچ‌های مختلف، خروجی الگوریتم را به خروجی مورد انتظار نزدیک‌تر کند. این تابع هزینه در فضای پارامترها حرکت کرده و مقدار بهینه را به دست می‌آورد.

ساختار این مقاله به این شکل است که در بخش دوم مروری بر الگوریتم‌های مختلف در فرآیند بهینه‌سازی مدل‌های عمیق مورد بحث و بررسی قرار گرفته است. در بخش سوم روش پیشنهادی با ارائه یک الگوریتم بهینه‌سازی با عنوان "انطباقِ پیشِ رو" برای آموزش مدل‌های عمیق مورد بحث قرار گرفته است. در بخش چهارم نتایج بدست آمده از آزمایش عملکردِ الگوریتم پیشنهادی بر روی یک بانک داده‌ی استاندارد ارائه شده است. نتیجه گیری و پیشنهادات آتی نیز در بخش پنجم ارائه شده است.

الگوریتم‌های پایه در فرآیند بهینه‌سازی

در این بخش الگوریتم‌های پایه بهینه‌سازی مورد استفاده در یادگیری عمیق مورد بررسی قرار می‌گیرند.

الگوریتم SGD

گرادیان نزولی تصادفی (SGD) و انواع آن، پر استفاده‌ترین الگوریتم بهینه‌سازی در یادگیری ماشین و بطور خاص در یادگیری عمیق است. با متوسط‌گیری از گرادیان روی کوچک دسته‌هایی که بطور i.i.d[2] نمونه‌برداری شده می‌توان تخمینی بدون بایاس از گرادیان تابع هزینه بدست آورد [۱۷].

[1] Stochastic Gradient Descent
[2] Independent and Identically Distributed

پارامتر مهم الگوریتم SGD نرخ یادگیری ϵ_k است. در عمل نرخ یادگیری با عبور زمان و اجرای تکرارها، کاهش داده می‌شود. علت انجام این‌کار این است که تخمین‌گر SGD با انجام نمونه‌برداری تصادفی، نویزی وارد می‌نماید که حتی با رسیدن به کمینه محلی هم از بین نمی‌رود. بنابراین بمنظور حصول همگرایی باید به سمت صفر میل کند. این در حالی است که در گرادیان معمولی بدلیل عدم وجود عامل تصادفی، با رسیدن به نقطه بهینه، گرادیان واقعی صفر می‌شود و با یک نرخ یادگیری ثابت، همگرایی حاصل می‌شود. یک فرض کافی مناسب برای تضمین SGD مطابق رابطه ۱ است [۱۸].

$$\sum_{k=1}^{\infty} \epsilon_k = \infty, \qquad \sum_{k=1}^{\infty} {\epsilon_k}^2 < \infty. \tag{۱}$$

در عمل، می‌توان نرخ یادگیری را مطابق رابطه ۲، بطور خطی از ϵ_0 در ابتدای کار، تا ϵ_τ در تکرار τ ام، تغییر داده و از تکرار τ ام به بعد ثابت نگاه داشت.

$$\epsilon_k = (1-\alpha)\epsilon_0 + \alpha\,\epsilon_\tau, \quad \alpha = \frac{k}{\tau} \tag{۲}$$

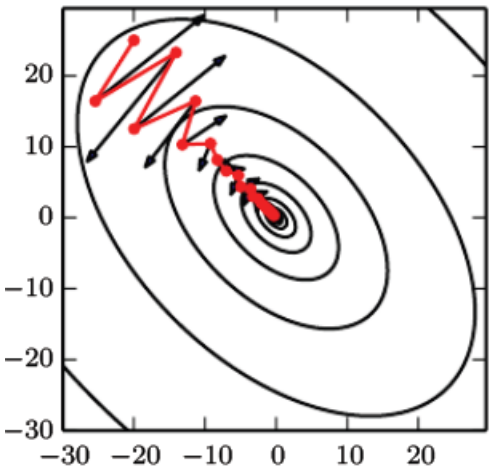
نرخ یادگیری را می‌توان بصورت سعی و خطا انتخاب کرد. روش بهتر این است که منحنی یادگیری رسم شود و تابع هدف را بصورت تابعی از زمان مانیتور کنیم. البته این کار باید با اندکی احتیاط انجام شود و بیشتر یک رویکرد هنری محسوب می‌شود تا علمی. بهنگام استفاده از رابطه ۲ پارامترهایی که باید انتخاب شوند ϵ_0، ϵ_τ و τ می‌باشند. معمولا τ به تعداد تکرارهای لازم برای چند صد بار عبور از مجموعه داده‌های آموزشی تنظیم می‌شود. همچنین ϵ_τ به مقداری در حدود یک صدم ϵ_0 تنظیم می‌شود. سوال اصلی که باقی می‌ماند مقدار ϵ_0 است. اگر مقدار ϵ_0 بزرگ انتخاب شود، منحنی یادگیری، نوسانات شدیدی را تجربه کرده و مقدار تابع هدف غالباً افزایش می‌یابد. البته نوسانات ملایم طبیعی است، مخصوصاً هنگامی که از تکنیک حذف تصادفی مانند dropout استفاده شود. از طرف دیگر اگر مقدار ϵ_0 کوچک انتخاب شود، فرآیند یادگیری پیشرفت کندی داشته و ممکن است یادگیری در یک نقطه غیر بهینه متوقف شود. برای تعیین ϵ_0 در ابتدا، ۱۰۰ تکرار اول با نرخ یادگیری‌های مختلف انجام می‌شود و عملکرد فرآیند یادگیری برمبنای تعداد تکرارها و حداقل تابع هزینه، ارزیابی شده و بهترین مقدار برای ϵ_0 بدست می‌آید. البته تجربه نشان داده که مناسب‌ترین انتخاب برای ϵ_0، مقداری بزرگتر از بهترین مقدار ϵ_0 بدست آمده از این روش است. مهمترین خاصیت الگوریتم بهینه‌سازی SGD، عدم وابستگی زمان محاسبات به تعداد نمونه‌های آموزشی است. به این ترتیب حتی با مجموعه داده‌های بسیار زیاد هم همگرایی امکان‌پذیر است. تحت این شرایط، این امکان وجود دارد که قبل از پردازش کل داده‌های آموزشی، خطای مدل، در یک محدوده مشخص از خطای تست نهایی قرار گیرد. بمنظور مطالعه نرخ همگرایی یک الگوریتم بهینه‌سازی، اندازه‌گیری خطای فزونی[1] $J(\theta) - \min_{\theta} J(\theta)$ عملی رایج است. خطای فزونی فاصله بین مقدار کنونی و مقدار کمینه تابع هزینه را نشان می‌دهد. هنگامی که الگوریتم SGD به یک مساله محدب و محدب قوی اعمال می‌شود، خطای فزونی بعد از k تکرار به ترتیب در مرتبه $O(\frac{1}{\sqrt{k}})$ و $O(\frac{1}{k})$ است. از دیدگاه تئوری، الگوریتم گرادیان نزولی نسبت به نسخه تصادفی آن، نرخ همگرایی بهتری دارد. با این حال کران کرامر-رائو[2] بیان می‌کند کران خطای عمومیت‌پذیری نمی‌تواند از $O(\frac{1}{k})$ سریعتر کاهش پیدا کند. در [۱۸] بحث شده که در یادگیری ماشین، دنبال کردن

[1] Excess Bound
[2] Cramer-Rao Bound

الگوریتم‌هایی با نرخ همگرایی سریعتر از $O(\frac{1}{k})$ بی‌فایده است و همگرایی سریعتر متناظر با وقوع بیش‌برازش[1] خواهد بود. در شرایطی که مجموعه داده‌هایی بزرگ در اختیار باشد، الگوریتم SGD قادر است با محاسبه گرادیان، تنها برای تعداد اندکی از نمونه‌ها، پیشرفت سریعی در ابتدای کار داشته باشد، بطوری که همگرایی حَدّیِ کُند آن به نوعی جبران می‌شود. بیشتر الگوریتم‌هایی که در ادامه توضیح داده می‌شوند، در عمل نتایج بهتری ارائه می‌دهند ولی در عین حال در تحلیل حدی، بصورت عامل ثابت تغییر ایجاد می‌کنند و تاثیری در مرتبه همگرایی ندارند. در عمل می‌توان با بزرگ کردن تدریجی دسته‌ها، مصالحه‌ای بین منافع الگوریتم‌های گرادیان نزولی و گرادیان نزولی تصادفی برقرار کرد.

الگوریتم Momentum

اگرچه الگوریتم SGD یک استراتژی بهینه‌سازی فراگیر است، با این حال در خیلی از مواقع به‌کُندی عمل می‌کند. الگوریتم Momentum (مومنتم) [۱۲] با هدف سرعت بخشیدن به فرآیند یادگیری، بخصوص در مواجهه با سطوح با انحنای شدید، سطوح با شیب ملایم و گرادیان نویزی، طراحی شده است. در این روش از گرادیان‌ها در قدم‌های قبلی، میانگین متحرک[2] با میرایی نمایی گرفته می‌شود و در این جهت، حرکت ادامه پیدا می‌کند. اثر اجرای الگوریتم Momentum در شکل ۱ نشان داده شده است. در این شکل پیکان‌های به رنگ مشکی جهت گرادیان و مسیر قرمز رنگ، مسیری است که الگوریتم Momentum طی می‌کند. مشاهده می‌شود که مسیر گرادیان تمایل دارد در عرض دره حرکات زیگ‌زاگی انجام دهد، درحالی که استفاده از Momentum باعث می‌شود مسیر حرکت تمایل بیشتری در جهت طول دره پیدا کند.



شکل ۱. اصلاح جهت گام‌های برداشته شده در مواجهه با وضعیت نامساعد در روش Momentum [۱۲].

الگوریتم Nestrove

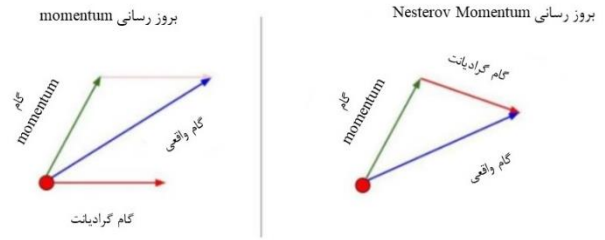
الگوریتم Nestrove یا نسترو [۱۳] از الگوریتم گرادیان تسریع شده الهام گرفته شده است. قانون به روزرسانی این روش جدید در عبارت ۳ نمایش داده شده است.

[1] Over Fitting
[2] Running Average

$$\upsilon \leftarrow \alpha\,\upsilon - \epsilon\,\nabla_{\theta}\left[\frac{1}{m}\sum_{i=1}^{m} \mathrm{L}\big(f(x^{(i)};\theta+\alpha\upsilon), y^{(i)}\big)\right] \qquad (۳)$$

$$\theta \leftarrow \theta + \upsilon$$

در این عبارت پارامترهای α و ϵ نقشی مشابهی با الگوریتم Momentum دارند. تفاوت دو الگوریتم Momentum و Nestrove در جایی است که گرادیان محاسبه می‌شود. همانطور که در شکل ۲ نشان داده شده، گرادیان بعد از اعمال سهم سرعت فعلی محاسبه می‌گردد. بعبارت دیگر می‌توان الگوریتم Nestrove را تلاشی در جهت اصلاح محل محاسبه گرادیان توصیف نمود. به این ترتیب در این الگوریتم یک نگاه روبه جلو[1] وجود دارد و سعی می‌شود گرادیان در نقطه‌ای که در قدم بعدی بطور حدودی در آنجا خواهد بود، محاسبه شود.



شکل ۲. محاسبه گرادیانِ الگوریتم Nestrove در انتهای بردار سبز رنگ بجای نقطه قرمز.

الگوریتم AdaGrad

محققین حوزه شبکه‌های عصبی دریافته‌اند که تنظیم مناسب فراپارامتر نرخ یادگیری یکی از سخت‌ترین‌ها کارهاست. عموما در هر تکرار، مقدار تابع هزینه در بعضی از جهت‌ها حساس و به بعضی دیگر حساس نیست. این ویژگی منجر به حرکات زیگ‌زاگی می‌شود. الگوریتم Momentum که برای مواجهه با این پدیده معرفی شد خود یک فراپارامتر جدید اضافه می‌کند. راه حل طبیعی پدیده حرکات زیگ‌زاگی، استفاده از نرخ یادگیری جداگانه برای هر پارمترها است. در الگوریتم AdaGrad هر پارامتر، نرخ یادگیری مربوط به خود را دارد و متناسب با مجذور مجموع تاریخچه مشتق‌های جزئی گذشته، تغییر مقیاس داده می‌شود [۱۴]. در این الگوریتم ترتیب نرخ یادگیری برای پارامترهایی که دارای سابقه مشتق‌های جزئی بزرگی هستند، بسرعت کاهش داده شده و برای پارامترهایی که دارای سابقه مشتق‌های جزئی کوچکی هستند، کاهش حداقلی را تجربه می‌کند. در مجموع الگوریتم در مواجهه با جهت‌هایی که شیب ملایم دارند سرعت پیمایش بهتری پیدا می‌کند. این الگوریتم در حوزه توابع هزینه محدب، خواص تئوری مطلوبی دارد. اما در عمل، با بکارگیری این روش در شبکه‌های عمیق و تقسیم نرخ یادگیری بر تجمیع کل تاریخچه مشتقات جزئی مشاهده شده، در بعضی مواقع، نرخ یادگیری بصورت زود هنگام قبل از رسیدن به نقطه بهینه، بیش از حد کوچک می‌شود.

الگوریتم RMSProp

الگوریتم RMSProp [۱۵] تجمیع گرادیان در الگوریتم AdaGrad را بصورت میانگین‌گیری متحرک با وزن‌دهی نمایی انجام می‌دهد. به این ترتیب این الگوریتم بهنگام استفاده در توابع هزینه غیر محدب می‌تواند سابقه گرادیان‌های

[1] Lookahead

دور را فراموش کرده و در میانه راه براحتی از سراشیبی‌ها به پایین حرکت کند. به الگوریتم RMSProp در مقایسه با AdaGrad، یک فرا پارامتر جدید اضافه شده که طول متوسط میانگین‌گیری حرکتی را تعیین می‌کند. در عمل الگوریتم RMSProp نتایج خوبی در آموزش مدل‌های عمیق ارائه داده است و امروزه به‌عنوان یکی از روش‌های متداول مورد استفاده قرار می‌گیرد.

الگوریتم Adam

الگوریتم Adam [۱۶] الگوریتمی دیگر با نرخ یادگیری وفقی است. کلمه Adam از عبارت "Adaptive Moments" مشتق شده است. این الگوریتم را می‌توان ترکیبی از دو الگوریتم Momentum و Adam در نظر گرفت. اول اینکه در این الگوریتم بطور مستقیم از مُمان مرتبه اول گرادیان، با وزن‌دهی نمایی استفاده شده و سرراست‌ترین راه برای اضافه کردن Momentum به RMSProp، اعمال کردن Momentum به گرادیان مقیاس شده است. دوم اینکه در اینجا تصحیح بایاسی بر تخمین‌های ممان مرتبه اول و دوم اعمال می‌شود. در الگوریتم RMSProp از تخمین مُمان مرتبه دوم استفاده می‌شود، البته بدون عامل تصحیح کننده، که موجب بالا رفتن بایاس در اوایل فرآیند یادگیری شود. الگوریتم Adam به‌عنوان الگوریتمی شناخته می‌شود که نسبت به انتخاب فراپارامترها مقاوم عمل می‌کند.

در این بخش، الگوریتم‌های پایه در فرآیند بهینه‌سازی برای یادگیری عمیق به طور خلاصه توضیح داده شد. مزایا و معایب این الگوریتم‌ها در جدول ۱ ارائه شده است. در این الگوریتم‌ها استخراج ویژگی‌های بهتر، مستلزم استفاده از تبدیل‌ها و روش‌های بسیار پیچیده‌تر است که باعث کند شدن روند بازیابی نیز می‌شود، اما عمل همگشت که مبنای اصلی مدل پیشنهادی و ترکیبی از عمل ضرب و جمع است، به مراتب ساده‌تر از محاسبات سایر روش‌ها مانند استفاده از تبدیل فوریه، تبدیل موجک و غیره می‌باشد. بنابراین سرعت بازیابی در این مدل نیز افزایش می‌یابد که از دیگر مزیت‌های استفاده از الگوریتم بهینه‌سازی پیشنهادی است.

جدول ۱. مقایسه مزایا و معایب الگوریتم‌های پایه در فرآیند بهینه‌سازی.

نام الگوریتم	مزایا	معایب
SGD	عدم وابستگی به داده‌های آموزشی	پایین بودن سرعت همگرایی
Momentum	افزایش سرعت همگرایی	گام‌های نادقیق در شیب‌های تند
Nestrove	گام‌های دقیق در شیب‌های تندتر	کاهش دقت در محاسبه گرادیان‌های محلی
RMS Prop	ایجاد نرخ یادگیری برای هر پارامتر	ایجاد گام‌های کوچک و هم اندازه در شیب تند
Ada Grad	ایجاد گام‌هایی با اندازه متفاوت	نداشتن بایاس دقیق در محاسبه گرادیان
Adam	تخمین ممان مرتبه اول و اصلاح بایاس	نرخ یادگیری بزرگ و تنزل گام‌ها

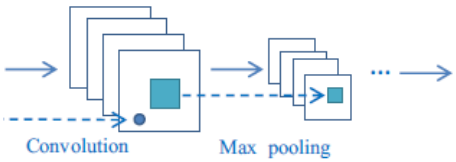
الگوریتم پیشنهادی "انطباقِ پیشِ رو"

اخیرا مدل‌های قدرتمند مبتنی بر یادگیری عمیق به منظور پردازش تصاویر ابداع شده‌اند. در روش‌های پردازش تصویر که تاکنون مطرح شده، اکثرا از ویژگی‌های سطح پایین تصاویر برای استخراج ویژگی استفاده می‌شود. به طور دقیق‌تر در این روش‌ها، ویژگی‌های استفاده شده به صورت دستی استخراج می‌شوند. بنابراین باز هم یکی از معایب مشترک این روش‌ها این است که فاصله‌ی معنایی میان ادراک انسان و این ویژگی‌ها مشهود است و دقت بازیابی را پایین می‌آورد. امروزه توجه محققان به روش‌هایی که این فاصله را کم کند معطوف شده است. یکی از روش‌هایی که در کاهش این فاصله موثر بوده است، یادگیری عمیق است. این روش به جای استفاده از ویژگی‌های استخراج شده‌ی دستی، به صورت خودکار و طی معماری سلسله مراتبی، ویژگی‌های مناسب و دقیق را از تصاویر استخراج می‌کند. در این مطالعه،

استفاده از یادگیری عمیق را نسبت به روش‌های مرسوم برگزیدیم. برای آموزش و پیاده‌سازی مدل‌های مبتنی بر شبکه‌های عصبی عمیق، سه راه پیش رو داریم. اول این که می‌توانیم یک شبکه را از اول تعریف کنیم و روی یک بانک داده‌ی استاندارد آموزش دهیم. این روش در صورتی نتیجه‌ی مطلوب می‌دهد که بانک داده‌ی تصاویر آموزشی شامل تعداد عظیمی از تصاویر باشد. بدیهی است به دلیل این‌که تعداد پارامترهایی که نیاز به آموزش دارند در شبکه‌های عمیق زیاد است، هر چه تعداد تصاویر ورودی بیش‌تر باشد، آموزش بهتر انجام می‌شود. روش دوم، استفاد از مدلِ از قبل آموزش داده شده است. این کار در صورتی بهینه است که تفاوت بانک داده‌ی مورد نظر و داده‌ی اولیه که مدل تحت آن آموزش دیده است، اندک باشد. روش سوم نیز تنظیم دقیق[1] نام دارد و به معنی استفاده از وزن‌های از پیش آموزش دیده شده برای تعداد زیادی از لایه‌ها و آموزش تعداد کمتری از لایه‌ها بر روی پایگاه داده‌ی مورد نظر است. ما برای ادامه‌ی کار، روش اول را برگزیدیم. معماریِ مدل عمیق ارائه شده در این مطالعه یک مدلِ مبتنی بر شبکه عصبی همگشتی (CNN) می‌باشد که برای بررسی عملکرد الگوریتم پیشنهادی ارائه شده است. این معماری از پنج لایه همگشت با پنجره‌های ۳ × ۳ و یک لایه تماما متصل[2] در خروجی، تشکیل شده است. در فرآیند آموزش مدل تعداد نورون‌ها در هر لایه به ترتیب برابر با ۷۰، ۶۰، ۵۰، ۵۰ و ۵۰، اندازه بسته[3]ها که اشاره به تعداد داده‌های آموزشی برای به‌روزرسانی در فرآیند بهینه‌سازی دارد، برابر با ۲۵۶ و تعداد ایپاک[4]ها، هشت در نظر گرفته شده است. این تصمیم با توجه به حجم داده‌های در دسترس و آزمایش‌های انجام شده، صورت گرفته است. در لایه‌ی اولِ معماری پیشنهادی به منظور حفظ اطلاعات مفید از عمل ادغام یا پولینگ[5] استفاده می‌شود. در لایه دوم و سوم عمل نرمالیزاسیون[6] به منظور بهبود عملکرد الگوریتم بهینه‌سازی، بصورت برخط انجام می‌شود و به منظور ایجاد رده‌های خروجی و تعریفِ تابع هدف یک لایه سافت‌مکس[7] در انتها به مدل اضافه می‌گردد. این لایه، هر پیکسل حجمی[8] را به تعداد کلاس‌های خروجی تقسیم می‌کند.

در این بخش، هدف ما ارائه‌ی معماری است که بتوانیم از شبکه‌های عصبی همگشتی عمیق به منظور قطعه بندی استفاده کنیم. در هر معماری همگشتی چندین لایه‌ی همگشت وجود دارد. در این لایه‌ها تصویر ورودی با فیلترهایی که ضرایب آنها قابل آموزش است، کانوالو (ضرب نقطه به نقطه) می‌شود. این فیلترها روی تصویر حرکت‌داده می‌شوند. عمق فیلتر با عمق تصویر برابر است. به ازای هر فیلتر مجزا، یک صفحه‌ی ویژگی ایجاد می‌شود. اگر از n فیلتر استفاده کنیم، n صفحه‌ی ویژگی به وجود می‌آید. وزن‌های هر فیلتر که w نامیده می‌شوند، قابل آموزش هستند و در طول آموزش شبکه، مرتبا به‌روز می‌شوند. بعد از عمل کانوالو، حاصل با یک عدد بایاس جمع شده و در صفحه‌ی ویژگی ذخیره می‌شود. به منظور تحقق بازیابی دقیق، معماری پیشنهادی بدین صورت است که ابتدا مدل را بر روی پایگاه-داده‌ی مورد نظر، تنظیم کنیم. برای این منظور وزن‌های لایه‌ها را تنظیم می‌کنیم و برای آموزش، همه‌ی تصاویر را از این شبکه‌ی عمیق عبور می‌دهیم. هر تصویر ابتدا با تعدادی فیلتر با اندازه‌ی ۳×۳ با مقدار دهی اولیه‌ی گوسی همگشت می‌شود به نحوی که فیلتر روی همه‌ی قسمت‌های تصویر حرکت داده می‌شود و حاصل همگشت، در صفحات ویژگی ذخیره می‌شود. پس از اعمال تابع فعال ساز به این صفحات، لایه اول مطابق شکل ۳ وارد مرحله‌ی ادغام ماکزیمم (Max Pooling) می‌شود.

[1] Fine Tune
[2] Fully Connected
[3] Batch
[4] Epoch
[5] Pooling
[6] Normalization
[7] Softmax
[8] Voxel



شکل ۳. نمایش کرنل‌های لایه اول (همگشت و ادغام) برای شبکه همگشتی عمیق پیشنهادی.

در مرحله‌ی ادغام ماکزیممی به نحوی پنجره‌ها روی تصویر حرکت داده می‌شوند که از میان پیکسل‌ها، ماکزیمم آن‌ها انتخاب شود. بنابراین پس از این لایه، اندازه‌ی صفحات ویژگی کاهش می‌یابد. قطعه کد مربوط به این بخش به ترتیب زیر می‌باشد:

```
input_img = Input (shape = (32, 32, 4)
x = Conv2D (70, (3, 3), activation='relu', padding='same') (input_img)
x = MaxPooling2D ((2, 2), padding='same') (x)
```

در این قطعه کد دستور shape برای ساختن ماتریس اندازه نورون‌ها، Conv2D برای ساختن همگشت‌های دو بعدی و MaxPooling برای ساختن ادغام از نوع ماکسیمم می‌باشد. دستور activation مربوط به انتخاب نوع تابع غیر خطی و padding برای پر کردن لایه‌های اطراف نواحی لبه استفاده می‌شود؛ که انتخاب نوع same باعث استفاده از اطلاعات آخرین خطوط اطراف لبه‌ها می‌باشد. در خط دوم قطعه کد بالا (input_img) به عنوان ورودی لایه و در خط سوم (x) بدست آمده از لایه‌ی قبل به عنوان ورودیِ این لایه مورد استفاده قرار می‌گیرد.

به منظور یافتن مقادیر بهینه برای پارامترهای مدل، یک الگوریتم بهینه‌سازی با نام "انطباقِ پیشِ رو" یا "Adaptation Ahead" پیشنهاد شده است. نام "Adaptation Ahead" برای الگوریتم بهینه‌سازی پیشنهادی انتخاب شده است، زیرا که با تعریف یک تابع هدف برای الگوریتم، سعی می‌شود مقادیر پارامترهای قابل تنظیم بگونه‌ای محاسبه گردند که تابع هدف مورد نظر، با توجه به مجموعه داده‌های آزمایشی کمینه شود. قطعه کد زیر بخشی از فرآیندِ مربوط به ساختِ مدل و انتخاب الگوریتم مورد نظر برای بهینه‌سازیِ مدل می‌باشد:

```
DCNN = Model (input_img, x)
DCNN.compile (optimizer= Adaptahead, loss='binary_crossentropy')
```

در قطعه کد بالا دستور Model مربوط به ساختن مدل با انتخاب input_img به عنوان لایه ورودی و x نهایی به عنوان لایه خروجی تعریف می‌شوند. بعد از ساختن مدل می‌بایست آن را کامپایل کنیم. از دستور Compile برای اجرا و کامپایل مدل استفاده می‌شود که پارامترهای مورد نظر به ترتیب مربوط به انتخاب الگوریتم بهینه‌سازی و تابع هزینه می‌باشند. از پارامتر optimizer برای انتخاب تابع بهینه سازی و از پارامتر loss برای انتخاب تابع هزینه استفاده می‌کنیم؛ که در این کار پارمتر "AdaptAhead" به عنوان الگوریتم بهینه‌سازی و تابع binary_Crossentropy به عنوان تابع هزینه انتخاب می‌شوند. بعد از انجام پیش پردازش‌های N4ITK و Nyul روی تصاویر ورودی، فراپارامترهای مدل از قبیل نرخ یادگیری، ضریب مومنتم، ضریب واریانس، مقداردهی اولیه می‌شوند. سپس گرادیان تابع هدف، بر اساس بسته داده آموزش ورودی محاسبه می‌شود. در قدم بعدی واریانس گرادیان محاسبه شده و با بکارگیری روش پولاک-ریبری[1] نرمالیزه می‌شود، سپس مومنتم محاسبه و نرمالیزه می‌شود. به این ترتیب در این روش از اطلاعات مرتبه اول و دوم

[1] Polak-Ribiere

توامان استفاده می‌شود. در نهایت، بردار پارامترهای مدل بصورت گرادیان نزولی به روزرسانی می‌شود. علاوه بر این نرخ یادگیری سراسری هم مطابق یک زمانبندی با گذر زمان کاهش می‌یابد. لازم به ذکر است که در فرآیند آموزش باید نظارت پیوسته‌ای بر دقت عملکرد مدل برای داده‌های اعتبارسنجی وجود داشته باشد و برای جلوگیری از بیش‌برازش با مشاهده عدم بهبود دقت در یک ایپاک، فرآیند آموزش، خاتمه داده شود. قطعه کد زیر مربوط به شروع فرآیند آموزش مدل می‌باشد:

```
n_epocks = 8
n_batch_size = 256
DCNN.fit (
        x_train,
        epochs = n_epochs,
        batch_size = n_batch_size,
        shuffle = True,
        validation_data = (x_test_val)
    )
```

در قطعه کد بالا قبل از ورود به فاز آموزشِ مدل، ابتدا تعداد گام یا ایپاک‌ها و سپس اندازه دسته‌ها را مشخص می‌کنیم. ایپاک‌ها تعداد مراحلی هستند که در هر کدام از آنها مدل تمام داده‌ها را می‌بینند. تعداد ایپاک‌ها هشت و اندازه دسته‌ها ۲۵۶ در نظر گرفته شده، سپس فرآیند آموزش مدل با دستور fit آغاز می‌شود. از پارامتر shuffle برای تغییر چینش داده‌ها و در واقع بُر خوردن داده‌ها استفاده می‌شود. True کردن مقدار این پارامتر باعث می‌شود مدل از تاثیر دادنِ توالی داده‌ها در نتیجه جلوگیری کند. پارامتر آخر Validation مربوط به داده‌های ارزیابی می‌باشد که در انتهای هر دور آموزش، داده‌های (x_test_val) که مربوط به داده‌های ارزیابی باشد را به مدل می‌دهیم تا در پایان هر دور آموزش نتایجش را بر روی این داده‌ها نیز عرضه کند و شاهد نتایج واقعی‌تر باشیم. در ابتدای الگوریتم بهینه‌سازی "انطباقِ پیشِ رو" قبل از ورود به حلقه اصلی، ثابت‌های الگوریتم تنظیم می‌شوند. پارامترهایی که در الگوریتم بهینه‌سازی پیشنهادی مورد استفاده قرار می‌گیرند در جدول ۲ معرفی شده است.

جدول ۲. معرفیِ متغیرهای مورد استفاده در الگوریتم بهینه‌سازی پیشنهادی "انطباقِ پیشِ رو".

نام متغیر	عملکرد متغیر
ϵ	نرخ یادگیری سراسری
λ_m	ممان اول
λ_v	ممان دوم
α	نرخ یادگیری اصطحکاک
ρ	مقدار نُرم
θ	تابع هدف
N	تعداد نورون‌ها (پارامترهای مدل)
t	اندیس تکرار
m	پارامتر مومنتوم
v	سرعت حرکت در فضای پارامترها
g	محاسبه گرادیان
$\triangle \theta$	محاسبه نرخ یادگیری (بروزرسانی)

جزئیات و مراحل مربوط به الگوریتم بهینه‌سازی پیشنهادی "انطباقِ پیشِ رو" در جدول ۳ نشان داده شده است. ϵ به‌منظور پایداری عددی اضافه شده است و از صفر شدن مخرج کسر در گام (سطر) ۱۲ جلوگیری به عمل می‌آورد. λ_m ثابت محاسبه ممان اول است که به منظور ایجاد یک پنجره لغزان نسبتا کوچک برای محاسبه ممان متوسط، به مقدار ۰/۹ تنظیم می‌شود. λ_v ثابت محاسبه ممان دوم است که به منظور ایجاد یک پنجره لغزان نسبتا طویل برای محاسبه ممان متوسط، به مقدار ۰/۹۹۹ تنظیم می‌شود؛ سپس به پارامترها یا همان متغیرهای اصلی مدل، مقدار اولیه داده می‌شود. نرخ یادگیری در الگوریتم ارائه شده در ابتدا به مقدار ۰/۰۱ تنظیم اولیه می‌شود و در ادامه بصورت بازپختی تغییر می‌کند. به عبارت دیگر در تکرارهای الگوریتم با مشاهده ثابت شدن نرخ خطا در داده‌های اعتبارسنجی مقدار نرخ یادگیری در ۰/۱ ضرب می‌شود. عدد نرم، فراپارامتر الگوریتم است و در سطر ۴ تنظیم می‌گردد. مقادیری که برای این فراپارامتر در نظر گرفته می‌شود ۱، ۲ و بینهایت است که متناظر با نرم ۱، نرم اقلیدسی و نرم ماکزیمم بوده و توسط سوئیچ ۱ (sw۱) انتخاب می‌شود. در یافته‌های تحقیق (بخش چهارم) برای هر یک از این مقادیر رفتار الگوریتم مورد ارزیابی قرار می‌گیرد. در گام ۵، وزن‌های مدل با مقادیر تصادفی گوسی با میانگین صفر و واریانس $2/N$ مقداردهی می‌شوند. در این رابطه N تعداد ورودی‌های نورون مربوطه می‌باشد. الگوریتم مادامی که نرخ خطا در داده‌های اعتبارسنجی سیر کاهشی دارد ادامه پیدا می‌کند. در خط ۹ سوئیچ ۲ (sw۲) تعیین می‌کند گرادیان به شیوه عادی یا Nestrov محاسبه شود. در سطر ۱۰ ممان اول محاسبه می‌شود که در واقع گرادیان را محاسبه می‌نماید. در سطر ۱۱ ممان مرتبه p محاسبه می‌شود. در سطر ۱۲سوئیچ ۳ (sw۳) تعیین می‌کند که نرخ یادگیری با بکارگیری ممان محاسبه شده در سطر ۱۲ به‌صورت وفقی عمل کند یا به شیوه معمولی بر مبنای Nestrov عمل کند. به این ترتیب با تنظیم سوئیچ‌های تعبیه شده، در کل می‌توان هشت شیوه ترکیبی مختلف را مورد بررسی قرار داد. در جدول زیر ملاحظه می‌شود با غیر فعال کردن نرخ یادگیری وفقی، نوع نرم بی‌تاثیر می‌شود.

جدول ۳. الگوریتم بهینه‌سازی پیشنهادی "انطباقِ پیشِ رو" (Adaptation Ahead).

گام‌های الگوریتم	مراحل الگوریتم
۱	ورود مجموعه داده‌های آموزشی شامل N نمونه
۲	تنظیم ثابت‌های الگوریتم: $\epsilon \leftarrow 10^{-8}, \lambda_v \leftarrow 0.999, \lambda_m \leftarrow 0.9$
۳	نرخ یادگیری بازپختی ابتدایی: $\alpha_0 = 0.01$
۴	عدد نرم مربوط به محاسبه واریانس: $p \in \mathbf{or}_{sw1}\{1, 2, \infty\}$
۵	مقداردهی اولیه پارامترها: $\theta_0 \leftarrow N(0, \sqrt{2/N})$
۶	مقداردهی اولیه صفر به ممان‌ها: $t, m_0, \upsilon_0 \leftarrow 0$
۷	while (کاهش خطا در داده‌های اعتبارسنجی)
۸	افزودن اندیس تکرار: $t = t + 1$
۹	محاسبه مومنتم بصورت میانگین متحرک: $m_t \leftarrow \lambda_m m_{t-1} + (1 - \lambda_m) g_t$
۱۰	محاسبه ممان بصورت میانگین متحرک: $v_t \leftarrow \lambda_v v_{t-1} + (1 - \lambda_v) \lvert g_t \rvert^{\mathbf{p}}$
۱۱	انجام یا عدم انجام نرمال‌سازی مومنتمی: $\Delta_t \leftarrow \left\{ \frac{1}{\sqrt[p]{v_t + \epsilon}} \ \mathbf{or}_{sw3}\ 1 \right\} \alpha_t m_t$
۱۲	انجام به‌روزرسانی پارامترهای مدل: $\theta_t = \theta_{t-1} - \Delta_t$
۱۳	end while

نتایج تجربی

در این بخش به منظور ارزیابی معماری و الگوریتم بهینه‌سازی "انطباقِ پیشِ رو"، با بکارگیری مجموعه داده کلتک ۲۵۶[1] مدل مبتنی بر شبکه عصبی همگشتی عمیق ارائه شده مورد آزمایش و بررسی قرار می‌گیرد.

محیط شبیه‌سازی

نمایش جزئیات مربوط به بستر نرم افزاری و سخت افزاریِ مورد نیاز برای پیاده‌سازیِ روش پیشنهادی در جدول ۴ نمایش داده شده است.

جدول ۴. پلتفرم پیاده‌سازی مدل پیشنهادی (بستر نرم‌افزاری و بستر سخت‌افزاری).

بستر نرم‌افزاری	بستر سخت‌افزاری
سیستم عامل: Linux Ubuntu	پردازنده گرافیکی (GPU): NVIDIA GeForce Titan ۱۲Gb x۳
زبان برنامه‌نویسی: Phyton	پردازنده مرکزی (CPU): INTEL XEON E۵-۲۶۲۰ (۲.۴GHZ, ۶ CORE, ۱۵) x۲
بستر کد نویسی: Caffe2	هارد HHD)): SSD ۱TB Samsung x۴ + MCP SSD ۲۵۰GB Samsung x۱ + MCP TB SATA Enterprise x۴۲
	منبع تغذیه: POWER ۱۲۰۰w

اینترفیس Python با نام pycaffe یکی از ماجول‌های Caffe است که اسکریپت[2]های آن را می‌توان در caffe/python یافت [۱۹]. برای انجام کارهایی مثل بارگذاری مدل‌ها، مدیریت I/O، بصری سازی شبکه[3] و حتی تغییر نحوه حل مدل کافیست Caffe را در کد پایتون خود وارد[4] کنیم که تمام داده‌های مربوط به مدل، مشتقات و پارامترهای آن برای خواندن و نوشتن در دسترس هستند [۲۰]. Caffe یکی از معروف‌ترین و همینطور یکی از سریعترین چارجوب‌ها برای توسعه برنامه‌ها در حوزه یادگیری عمیق است که در ابتدا توسط جیا[5] و همکاران در دوران تحصیلش در مقطع دکتری در دانشگاه برکلی[6] توسعه داده شد [۱۹]. در حال حاضر Caffe بصورت متن باز[7] انتشار داده شده و توسط مرکز آموزش و بینایی دانشگاه برکلی ایالات متحده آمریکا توسعه داده می‌شود. در Caffe مدل‌سازی و بهینه‌سازی‌ها توسط فایل پیکربندی بدون آنکه هیچ نیازی به برنامه‌نویسی آنها باشد انجام می‌شود. انتخاب بین اجرای یادگیری[8] توسط CPU

[1] Caltech 256
[2] Script
[3] Visualize Network
[4] Import
[5] Jia
[6] Berkeley
[7] Open Source
[8] Training

و یا GPU تنها با مشخص کردن یک فلگ[1] در فایل پیکربندی قابل انجام است؛ و بعد از آن مدل آموزش دیده براحتی قابل انتقال به سیستم‌های دیگر جهت اجرا خواهد بود. این چارچوب دائما در حال توسعه است. تنها در سال اول انتشار Caffe این پروژه توسط بیش از ۱۰۰۰ برنامه نویس (توسعه دهنده) استفاده شد و هرکدام دستاوردها و تغییرات قابل توجهی را در پروژه به ارمغان آوردند. این چارچوب از آخرین نتایج رخ داده در حوزه یادگیری عمیق هم در کد و هم در مدل‌ها پیروی می‌کند. بواسطه پیاده‌سازی بسیار بهینه و استفاده ازGPUها این چارچوب در حال حاضر سریعترین چارچوب توسعه با استفاده از شبکه‌های عصبی همگشتی می‌باشد. به‌عنوان مثال این چارچوب قادر به پردازش بیش از ۶۰ میلیون تصویر در روز با استفاده از یک کارت گرافیک GPU Nvidia K۴۰ می‌باشد. این یعنی تنها یک میلی ثانیه به ازای تشخیص هر تصویر (در زمان تست) و چهار میلی ثانیه به ازای آموزش هر تصویر (در زمان آموزش). در Caffe نیز همانند بیشتر مباحث موجود در یادیگیری ماشین، یادگیری از طریق یک تابع خطا میسر و هدایت می‌شود. یک تابع خطا با نگاشت تنظیمات پارامترها (یعنی وزن‌های فعلی شبکه) به یک مقدار عددی و مشخص کردن میزان "بد بودن" یا "نامناسب بودن" این پارامترها هدف یادگیری را مشخص می‌کند. بنابراین هدف یادگیری یافتن تنظیماتی برای این وزن‌هاست، بگونه‌ای که مقدار تابع خطا به حداقل برسد.

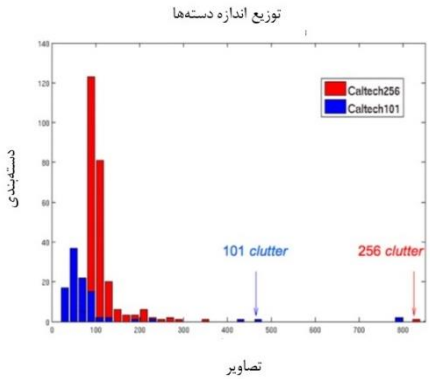
مجموعه داده‌های آموزشی

الگوریتم پیشنهادیِ بدست آمده، وابسته به داده‌های آموزشی نیست. برای ارزیابیِ عملکردِ الگوریتم بهینه‌سازی "انطباقِ پیشِ رو"، بر روی مجموعه داده‌های کلتک ۲۵۶ با معماریِ ارائه شده‌ی سه لایه در بخشِ قبل و با تعداد نورون‌های ۶۰ برای هر سه لایه مورد بررسی و آزمایش قرار گرفت. کلتک ۲۵۶ یک مجموعه داده از کلاس‌های مختلف تصویر مشابه شکل ۴ می‌باشد. این مجموعه داده شامل ۳۰۶۰۷ تصویر است که از هر تصویر نمونه‌های مختلفی در هر کلاس موجود می‌باشد.

شکل ۴. نمونه تصاویر مجموعه داده‌های کلتک ۲۵۶.

کلتک ۲۵۶ نمونه پیشرفته‌ای از کلتک ۱۰۱ می‌باشد. در کلتک ۱۰۱ حداقل نمونه‌های موجود در هر کلاس ۵۰ نموه بود، ولی همانطور که در نمودار شکل ۵ نشان داده شده، در کلتک ۲۵۶ بیشتر کلاس‌ها حداقل ۱۰۰ نمونه دارند.

[1] Flag

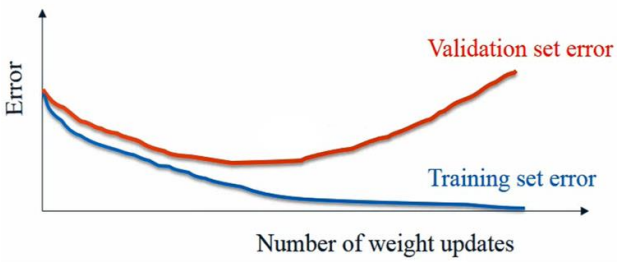


شکل ۵. نمایش تعداد نمونه‌های موجود در کلتک ۱۰۱ در مقایسه با کلتک ۲۵۶.

تجزیه و تحلیل مدل پیشنهادی

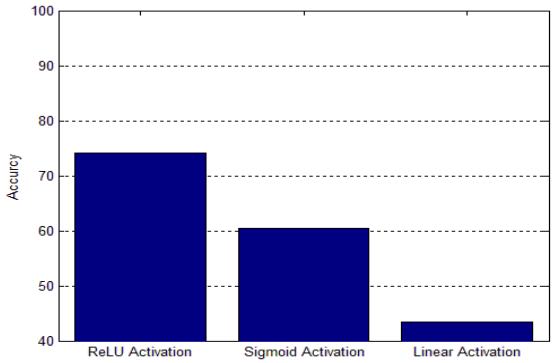
در این بخش نتایج آزمایش‌ها و انتخاب‌های مختلف برای شبکه پیشنهادی، از جمله تعداد نورون‌ها، اندازه دسته‌ها، اپیاک‌ها، لایه‌های ادغام، لایه‌های همگشت و اندازه پنجره همگشت مورد بحث و بررسی قرار می‌گیرد.

تعداد نورون‌ها: در این قسمت تعداد نرون‌های هر لایه همگشت بررسی می‌شود. با در نظر گرفتن تعداد ۲۵ برای نورون‌ها در همه لایه‌ها، دقت ۷۵ درصد حاصل می‌گردد. با افزایش تعداد نورون‌ها به ۵۰، دقت مدل به ۸۰ درصد می‌رسد. باید توجه داشت این نتایج بدون تنظیم دقیق فراپارامترها حاصل می‌شود. در ابتدا با افزایش تعداد نورون‌ها در لایه اول ظرفیت مدل افزایش یافته و سبب می‌شود ویژگی‌های مفید بیشتری در لایه اول شناسایی شود، ولی با ادامه افزایش تعداد نورون‌ها در لایه اول همانند شکل ۶ بیش‌برازش اتفاق می‌افتد؛ در واقع حجم داده‌ها به قدری افزایش می‌یابد که مدل در مرحله آموزش داده‌ها را حفظ می‌کند، و در این صورت با اینکه میزان خطا در مرحله آموزش کاهش می‌یابد ولی در مرحله اعتبارسنجی میزان خطا افزایش یافته و مدل نسبت به داده‌های جدید عملکرد مناسبی ارائه نمی‌دهد. با ادامه افزایش تعداد نورون‌ها تغییر محسوسی در دقت مدل حاصل نمی‌شود و فقط بار محاسباتی افزایش می‌یابد. بنابراین تعداد نورون‌ها در لایه‌های همگشت به ترتیب ۷۰، ۶۰، ۵۰، ۵۰ و ۵۰ نهایی شد. با توجه به تعداد کمتر پارامترها در سه لایه اخیر، کاهش نورون‌ها در این لایه‌ها منجر به تغییر کارایی در دقت می‌شود.



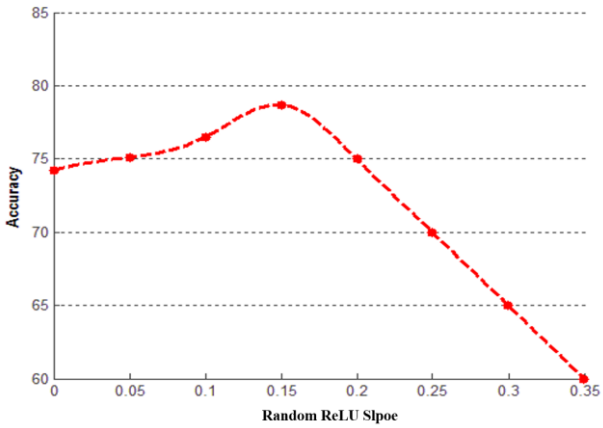
شکل ۶. نمایش بروز بیش برازش با افزایش تعداد نرون‌های لایه اول.

تابع فعال‌ساز: برای جلوگیری از اشباع نورون‌ها و به روز رسانی پارامترها، ReLU در همه لایه‌ها مورد استفاده قرار گرفت. تاثیر تابع غیر خطی ReLU بر دقت عملکرد مدل نسبت به دو تابع خطی و سیگموئید در شکل ۷ نمایش داده شده است.



شکل ۷. مقایسه استفاده از تابعِ خطی، سیگموئید و غیر خطی ReLU بر دقت عملکرد.

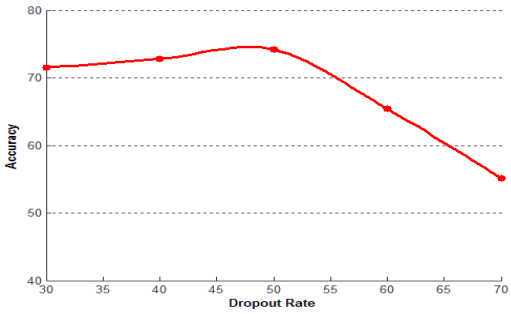
تابع غیر خطی ReLU دارای سه نوع ساده، پارامتریک و تصادفی می‌باشد که به دلیل استفاده از تایثر ناحیه منفی از نوع تصادفی استفاده شده است. همانطور که در شکل ۸ نشان داده شده است، در ابتدا از اطلاعات ناحیه منفی تا حدودی استفاده می‌شود، ولی در ادامه میزان غیر خطی بودن مدل رفته رفته کمتر می‌شود.



شکل ۸. نمایش تاثیر شیب ناحیه منفی تابع غیرخطی ReLU.

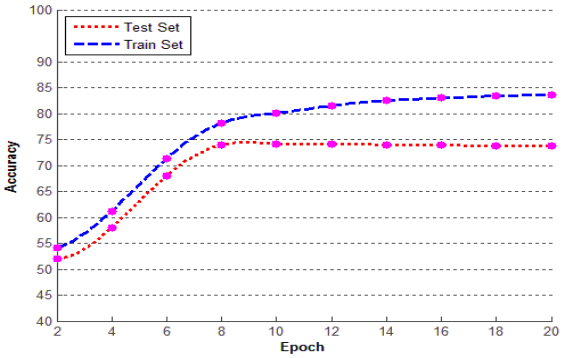
تکنیک حذف تصادفی[1]: علاوه بر این، برای جلوگیری از همگام‌سازی نورون‌ها و ایجاد طبقه‌بندی ترکیبی ضمنی، تکنیک حذف تصادفی استفاده شد. همانطور که در شکل ۹ نشان داده شده است، میزان استفاده ۵۰ درصدی از تکنیک حذف تصادفی در فرآیند آموزش باعث می‌شود همسازی نرون‌ها کاهش یافته و عمومیت پذیری مدل افزایش پیدا کند، ولی در ادامه به دلیل حذف اطلاعات مفید، موجب پیش‌برازش می‌شود و ظرفیت مدل به شدت افت می‌کند.

[1] Dropout



شکل ۹. نمایش تاثیر نرخ حذف تصادفی در لایه تماما متصل.

اندازه دسته و ایپاک‌ها: اندازه دسته اشاره به تعداد داده‌های آموزشی برای یک به‌روزرسانی در فرآیند بهینه‌سازی دارد. انتخاب اندازه دسته، بطور مستقیم بر هزینه محاسباتی و عدم قطعیت به‌روزرسانی تاثیر می‌گذارد. دسته‌های کوچک به‌روزرسانی نویزی‌تری ایجاد می‌کنند، اما در شرایطی که تابع خطا، کمینه‌های محلی زیادی دارند، وجود نویز در گرادیان باعث می‌شود، الگوریتم در نواحی که عمق کمی دارد گیر نکند. از طرف دیگر دسته‌های بزرگ باعث می‌شود عملیات بیشتری بصورت موازی انجام شود و این کمک می‌کند سرعت همگرایی بهبود پیدا کند. بنابراین باید در بدست آوردن اندازه دسته بهینه یک تعادل ایجاد شود. در اینجا با تغییر اندازه دسته از ۶۴ تا ۵۱۲، مقدار بهینه، ۲۵۶ بدست آمد. در آموزش مدل، تعداد ایپاک‌ها، هشت درنظر گرفته شد، زیرا با افزایش تعداد آنها، در دقت مدل تغییری ایجاد نمی‌شود. همانطور که در شکل ۱۰ نشان داده شده است بعد از ایپاک هشتم، به تدریج در مدل به میزان کمی بیش‌برازش رخ می‌دهد که با استفاده از حذف تصادفی میزان بیش برازش حداقل شده است.

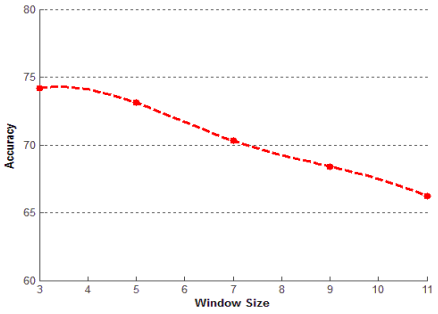


شکل ۱۰. نمایش تاثیر تعداد تکرارها بر دقت عملکرد مدل.

لایه‌های ادغام: در اینجا ادغام به منظور خلاصه‌سازی اطلاعات لایه قبلی، و بعد از همگشت و یکسوساز قرار داده شده است. با انجام آزمایشات، استفاده از لایه ادغام در لایه اول همگشت مفید واقع شد و استفاده از ادغام در لایه‌های بعدی منجر به از بین رفتن اطلاعات سودمند شد. ولی در لایه اول به دلیل همپوشانی همگشت‌ها، اطلاعات مفید حذف نمی‌شود.

لایه‌های همگشت: در ابتدا، کار با یک شبکه، با سه لایه همگشت و یک لایه تماما متصل شروع شد. دقتی که در این حالت بدست آمد ۸۶ درصد بود. با افزایش لایه‌ها به پنج لایه همگشت دقت به ۹۳ درصد رسید و با افزایش بیشترِ تعداد لایه‌ها، دقت ثابت ماند. همچنین با افزایش تعداد لایه‌های تماما متصل تغییری در دقت عملکر شبکه مشاهده نشد. بنابراین تعداد لایه بهینه با تعداد پارامترهای کمینه پنج لایه همگشت و یک لایه تماما متصل است.

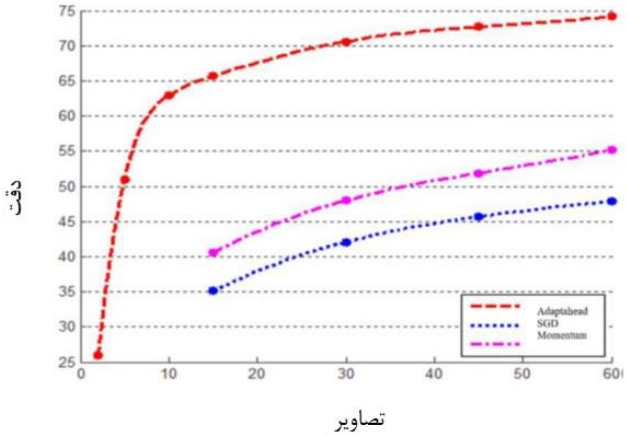
اندازه پنجره‌های همگشت: در آزمایشات انجام شده اندازه کرنل‌های سه، پنج و هفت بررسی شد و مشاهده گردید با افزایش ابعاد کرنل، دقت مدل کاهش می‌یابد. علاوه بر این مزیت دیگر استفاده از کرنل‌های کوچک، کاهش بار محاسباتی است. اندازه کرنل‌های همگشت در تمام لایه‌ها ۳×۳ در نظر گرفته شد. با افزایش اندازه پنجره، همانند شکل ۱۱ رفته رفته از حالت محلی بودن خارج شده و باعث می‌شود یک بهم ریختگی در محاسبه ویژگی‌های محلی ایجاد شود.



شکل ۱۱. نمایش تاثیر اندازه کرنل‌های همگشت.

نتایج خروجی

نتیجه اجرای الگوریتم بهینه‌سازی "انطباقِ پیشِ رو" بر روی مجموعه داده کلتک ۲۵۶ در مقایسه با دو الگوریتم SGD و Momentum در شکل ۱۲ نمایش داده شده است. نتایج بدست آمده نشان می‌دهد تنها با داشتن ۶ نمونه از هر کلاس می‌توان به دقتی که مدل‌های غیر عمیق با داشتن ۶۰ نمونه رسیده‌اند، دست یافت.



شکل ۱۲. نتیجه اجرای الگوریتم بهینه‌سازیِ پیشهادی بر روی مجموعه داده کلتک ۲۵۶.

برای ارزیابی عملکرد الگوریتم بهینه‌سازی "انطباقِ پیشِ رو" از معیار شباهت دایس [1]استفاده شده است. مدل پیشنهادی هر وکسل را به پنج رده تقسیم می‌کند. در این ارزیابی برای هر خروجی دو نقشه دودویی وجود دارد که یکی توسط مدل بدست می‌آید(P) و دیگری توسط اجماع متخصیص داده می‌شود (T) و با استفاده از خروجی مدل، معیار شباهت دایس مطابق رابطه ۴ محاسبه می‌شود. برای محاسبه دقت می‌توان از رابطه ۵ استفاده کرد که در این رابطه (A) نشان دهنده تمام نقشه دودویی است. این معیار به معنای نسبت ناحیه مشترک به متوسط ناحیه مشخص شده توسط مدل و متخصص است. در فرمول‌های زیر Λ به عنوان عملگر AND منطقی و $|\ |$ برای نشان دادن اندازه مجموعه (تعداد وکسل‌های متعلق به این مجموعه) استفاده می‌شوند. برای هر کلاس ما یک رده نقشه دودویی $P = \{0,1\}$ برای خروجی‌های مدل و یک رده نقشه دودویی $T = \{0,1\}$ برای خروجی‌هایی که توسط اجماع متخصصان در بانک داده بدست آمده داریم.

$$Dice(P,T) = \frac{|P \Lambda T|}{(|P| + |T|)/2} \quad (۴)$$

$$\text{Accuracy } (P,T) = \frac{|P \Lambda T| + |(A - P) \Lambda (A - T)|}{|A|} \quad (۵)$$

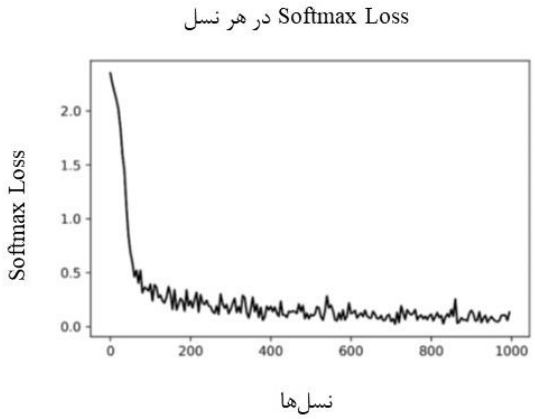
در جدول ۵ متناظر با انتخاب هر یک از سوئیچ‌ها، نتایج بدست آمده از آموزش مدل عمیق پیشنهادی ارائه شده است. استفاده از نرم دو در محاسبه ممان و بکارگیری تکنیک Nestrov در محاسبه گرادیان بهترین نتیجه یعنی دقت ۹۱/۱ را به همراه داشته است. از طرفی مشاهده می‌شود عدم استفاده از نرخ یادگیری وفقی و تکنیک Nestrov در محاسبه گرادیان منجر به عدم موفقیت در رسیدن به نقطه بهینه مناسب شده است. نکته قابل توجه دیگر آنکه استفاده از نرم‌های یک و بی‌نهایت در محاسبه ممان، اندکی کاهش در دقت نهایی به همراه داشته است.

جدول ۵. نتایج اجرای الگوریتم بهینه‌سازی "انطباقِ پیشِ رو" با انتخاب سوئیچ‌های مختلف.

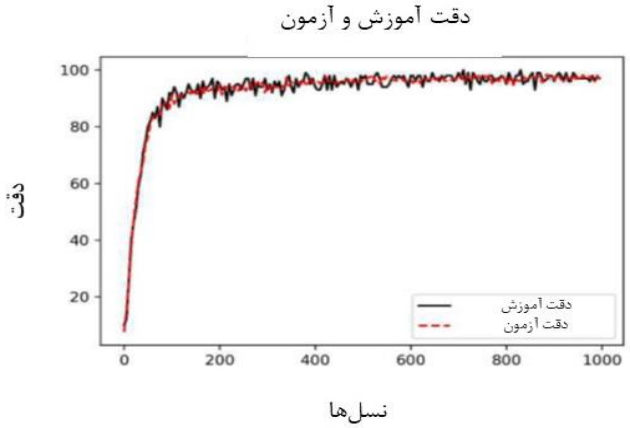
شماره حالت‌ها	انتخاب سوئیچ۱ SW1(Norm)	انتخاب سوئیچ ۲ SW2(Nestrov)	انتخاب سوئیچ ۳ SW3(Adaptive)	دقت بدست آمده از محاسبه معیار شباهت دایس $\text{Accuracy } (P,T) = \frac{\lvert P \Lambda T\rvert + \lvert (A - P) \Lambda (A - T)\rvert}{\lvert A\rvert}$
۱	-	Yes	No	۸۵/۱
۲	-	No	No	۸۳/۵
۳	۱	Yes	Yes	۸۷/۳
۴	۱	No	Yes	۸۶/۱
۵	۲	Yes	Yes	۹۱/۱
۶	۲	No	Yes	۸۸/۲
۷	∞	Yes	Yes	۸۶/۴
۸	∞	No	Yes	۸۵/۵

[1] Dice Similarity Coefficient Metric

علاوه بر اینکه استفاده از نرم دو در محاسبه ممان و بکارگیری تکنیک Nestrov در محاسبه گرادیان بهترین نتیجه یعنی دقت ۹۱/۱ را به همراه داشته است، از طرفی هم مشاهده می‌شود عدم استفاده از نرخ یادگیری وفقی و تکنیک Nestrov در محاسبه گرادیان منجر به عدم موفقیت در رسیدن به نقطه بهینه مناسب شده است. نکته قابل توجه دیگر آنکه استفاده از نرم‌های یک و بی‌نهایت در محاسبه ممان، اندکی کاهش در دقت نهایی به همراه داشته است.



شکل ۱۳. نمودار تغییرات تابع هزینه بر حسب تعداد تکرارها × ۱۰.



شکل ۱۴. نمودار تغییرات دقت مدل بر حسب تعداد تکرارها × ۱۰.

در شـــکل‌های ۱۳ و ۱۴ وجود نوســـان‌های نویزی در نمودار تابع هزینه و نمودار دقت مجموعه آموزشـــی به دلیل رفتارهای تصادفی کوچک دسته‌ها می‌باشد. ولی نمودار دقت مدل مجموعه تست در شکل ۱۴ نشان می‌دهد فرآیند بهینه‌سازی همگرا شده است. در مرحله آموزش شبکه، با توجه به اینکه می‌دانیم تصویر ورودی در چه کلاسی از کلاس‌های خروجی قرار دارد یک تابع هزینه تعریف می‌شـــود، به نحوی که میزان مرتبط بودن تصـــویر به آن کلاس مربوطه را اندازه بگیرد و هر چه این تابع برای یک کلاس کمتر از سایر کلاس‌ها باشد، حاکی از این است که تصویر به آن کلاس مرتبط است. وزن‌های فیلترها را طوری انتخاب می‌کنیم که این تابع کمینه شود. همانطور که از نمودار تابع

هزینه بر حسب تعداد تکرارها مشاهده می‌شود، مقدار تابع هزینه مرتباً کاهش پیدا کرده تا در حوالی ۱۰۰۰ تکرار تقریبا ثابت می‌شود، که نشان‌دهنده همگرا شدن مدل است. همچنین در نمودار دقت مدل برای داده‌های تست، برحسب تعداد تکرارها، مشاهده می‌شود که دقت مدل به تدریج افزایش یافته و در نهایت تقریبا ثابت می‌شود.

در نهایت برای مقایسه مدل ارائه شده با سایر منابع به روز، روش پیشنهادی بر روی تصاویر مربوط به MR مغزی نیز مورد بررسی قرار گرفت. میانگین دقتی که این مطالعه از نقطه نظر معیار شباهت دایس، برای سه حالت کامل[1]، هسته[2] و نواحی درحال پیشرفت[3] بر روی بانک تصویر BraTS۲۰۲۰ بدست آورده [۲۱] به ترتیب ۰/۹۳، ۰/۸۸ و ۰/۸۷ می‌باشد. این مجموعه داده شامل ۲۳۰ تصویر مغزی است. به منظور تشکیل مجموعه داده‌های مورد نیاز شبکه همگشتی عمیق، یک در میان پیکسل‌ها توموری و سه در میان پیکسل‌های سالم انتخاب شده و به همراه برچسب مربوطه ذخیره می‌شود. در اینجا، هدف قطعه‌بندی و بطور همزمان آشکارسازیِ زیرساختارها است که قطعه‌بندی، مبتنی بر تکه‌های ۳۲ × ۳۲ استخراج شده از تصاویر MR مغز انجام می‌شود. برچسب هر تکه تصویر، برابر با برچسب پیکسل مرکز هر تکه درنظر گرفته می‌شود و از چهار مدالیته، به‌عنوان چهار کانال مختلف برای هر تکه تصویر ورودی استفاده می‌شود. در نهایت هر پیکسل متناظر با میزان خطرناکی در پنج رده مختلف رده‌بندی می‌شود. این رده‌ها به ترتیب میزان خطرناکی از کمتر به بیشتر عبارتند از بافت نرمال (رنگ خاکستری)، مایع (رنگ سبز)، تومور غیر پیشرفته (رنگ آبی)، هسته تومور (رنگ قرمز) و تومور پیشرفته (رنگ زرد). جدول ۶ ماتریس سردرگمی[4] را نشان می‌دهد. ماتریس سردرگمی به عنوان یک جدول خاص موجب تجسم عملکرد یک الگوریتم یادگیری تحت نظارت می‌شود. از ماتریس سردرگمی برای تخمین برچسب پیکسل مرکزی تکه‌های تصویر MR مغزی استفاده می‌شود. هر سطر از این جدول نشان دهنده فاصله واقعی[5] برای نقاط مرکزی و هر ستون نشان دهنده فاصله کلاس پیش‌بینی[6] شده برای نقاط مرکزی است. به طور خاص در این جدول پنج سطر و پنج ستون وجود دارند که زیر ساخت‌های مختلف تومورهای گلیوبلاستوما (یعنی بافت نرمال، مایع، تومور غیر پیشرفته، هسته تومور و تومور پیشرفته) را مشخص می‌کنند. ماتریس سردرگمی نتایج آزمایش مدل همگشتی عمیق پیشنهاد شده را خلاصه می‌کند. به عبارت دیگر، هر پیکسل حجمی باید به یکی از پنج کلاس مشخص شده طبقه‌بندی شود.

جدول ۶. ماتریس سردرگمی برای تخمین برچسب پیکسل مرکزی تکه‌های تصویر MR مغزی.

پیش‌بینی / واقعی	بافت نرمال	مایع	تومور غیر پیشرفته	هسته تومور	تومور پیشرفته
بافت نرمال	**۵۲۳۵۵**	۵	۳۸۱	۷۶	۲۲۷
مایع	۱۳	**۶۰۱۳**	۹۸	۲۱۵	۹۳۱
تومور غیر پیشرفته	۱۳۶۰	۱۰۱	**۴۲۶۷۱**	۱۳۲۰	۱۰۲۳
هسته تومور	۴۰	۱۳۰	۳۶۵	**۸۴۴۷**	۷۲۵
تومور پیشرفته	۱۵۰	۲۵۰	۲۵۰	۲	**۱۰۷۰۹**

مقایسه مزایا و معایب قطعه‌بندی تصاویر MR مغزی با سایر منابع به روز نیز انجام شده و در جدول ۷ گزارش شد. درحالی که این روش‌های جدیدتر نسبت به روش‌های مرسوم گذشته بهبود بسیاری داشته است، بهبود بیشتر دقت

[1] Complete
[2] Core
[3] Enhancing Areas
[4] Confusion Matrix
[5] Actual
[6] Predicted

عملکرد و کاهش زمان آموزش و تست بسیار حائز اهمیت است. در [۲۲] برای قطعه‌بندی تصاویر MR مغزی از یک مدل آماده مبتنی بر U-Net استفاده شده است. در [۲۳] تشخیص و قطعه‌بندی تصاویر MR مغزی با استفاده از یادگیری عمیق و بهینه‌سازی گرگ خاکستری انجام شده است. در [۲۴] برای قطعه‌بندی تصاویر MR مغزی از مدل LDCRF[1] (میدان‌های تصادفی شرطی نهفته-دینامیک) استفاده شده است. در این روش برای بهبود فرآیند بهینه‌سازی از ویژگی‌های یک شبکه کپسول عمیق نیز استفاده شده است. در روش مورد استفاده در این مطالعه نیز از شبکه عصبی همگشتی (CNN) برای فرآیند قطعه‌بندی تصاویر MR مغزی استفاده شد.

جدول ۷. مقایسه کارهای انجام شده مربوط به سال‌های اخیر، در زمینه قطعه‌بندی تصاویر MR.

منبع	سال انتشار	روش کار	بانک تصاویر	معایب و چالش‌ها
[۲۲]	۲۰۲۳	U-Net	BraTS ۲۰۲۱	زمان بر بودنِ استفاده مستقیم از مدل برای قطعه‌بندی بافت تومور مغزی و سازگارکردن آن برای پردازش توسط U-Net
[۲۳]	۲۰۲۳	Grey Wolf	BraTS ۲۰۲۱	زمان تست و آموزش بالا به دلیل توابع فعالیت یکسوساز دارای نشت متوالی
[۲۴]	۲۰۲۲	LDCRF	BraTS ۲۰۲۱	زمان آموزش بالا در اعمال ایده تخمین ساختاری محلی
#	-	CNN	BraTS ۲۰۲۲	قابلیت انتزاع پایین شبکه‌های عصبی به دلیل استفاده از تنها روش CNN در قطعه‌بندی

نتیجه‌گیری و پیشنهادات آتی

عموما مدل‌های یادگیری از یک معماری و پارامترهای قابل تنظیم تشکیل می‌شوند. در این مطالعه یک مدل عمیق پیشنهاد شد. این مدل عمیق شامل دو بخش معماری و الگوریتم بهینه‌سازیِ پیشنهادی بود. معماری مربوط به ساختار و نحوه آرایش اجزای مدل است که یک طراحی بصورت تجربی ارائه می‌دهد و در مرحله بعد با تعریف یک تابع هدف، با استفاده از الگوریتم بهینه‌سازی سعی می‌شود مقادیر پارامترهای قابل تنظیم بگونه‌ای محاسبه شوند که تابع هدف مورد نظر، با توجه به مجموعه داده‌های آزمایشی کمینه شود. برای آموزش مدل پیشنهادی، تابع هزینه با وجود غیرخطی بودن باید به کمترین حد خود برسد که برای این کار از الگوریتم "انطباقِ پیشِ رو" یا "Adaptation Ahead" به عنوان الگوریتم بهینه‌سازی استفاده شد. الگوریتم گرادیان تصادفی نزولی پایه، متناسب با مقدار منفی گرادیان در جهت کمینه‌های محلی اقدام می‌کند، ولی با اینهمه در نواحی دارای انحنای کم ممکن است کند عمل نماید. برای تسریع الگوریتم بهینه‌سازی "انطباقِ پیشِ رو"، در نواحی کمینه‌های محلی از گشتاور شتابدار Momentum کمک گرفته شد. در این گشتاور v ثابت حفظ می‌شود، ولی سرعت یادگیری ε پس از هر دوره به صورت خطی کاهش یافته است. گرادیان‌های معمولی در صورت فعال نبودن واحد صفر هستند که این امر می‌تواند سرعت همگرایی را طی روند بهینه‌سازی کند کرده و بدترین نوع آموزش را به دنبال داشته باشد. ما برای جلوگیری از وقوع این مسئله ReLU را به عنوان حالت غیرخطی جایگزین کردیم. در روش‌های دیگر مهم‌ترین عامل کاهش دقت، استخراج ویژگی‌های سطح پایین تصاویر و عدم کاهش فاصله معنایی میان ادراک انسان و این ویژگی‌ها است. در این مطالعه با کمک یادگیری عمیق، استخراج سلسله مراتبی و عمیق ویژگی از تصاویر انجام شد. الگوریتم بهینه‌سازی "انطباقِ پیشِ رو"، که در آن از یک مدل عمیق مبتنی بر شبکه عصبی همگشتی استفاده شده، ویژگی‌های سطح بالاتری را استخراج کرده و به دقت مطلوبی دست یافته است. با بکارگیری تکنیک Nestrov در محاسبه گرادیان توسط الگوریتم پیشنهادی بهترین نتیجه یعنی دقت ۹۱/۱ برای معیار شباهت دایس بدست آمد. برتری دیگر این الگوریتم نسبت به سایر روش‌ها، استفاده از محاسبات

[1] Latent-Dynamic Conditional Random Fields

غیر پیچیده است. قابل توجه است که به هنگام محاسبه گرادیان برای یک دسته با بکارگیری GPU و موازی‌سازی داده‌ای و تابعی، می‌توان سرعت همگرایی به میزان ۱۰× افزایش داد.

در کارهای آینده می‌توان با در دست داشتن سخت افزارهای پیشرفته با حجم اطلاعاتیِ بالاتر، به‌منظور لحاظ کردن توام ویژگی‌های محلی و سراسری در تکه تصویر ورودی، سه مسیر کمکی ایجاد کرد و به همراه مسیر اصلی به لایه تماما متصل داد. در مسیر کمکی اول کرنل‌های ۱۵ × ۱۵ با پرش‌های ۱۵ تایی در نظر گرفت، در مسیر دوم و سوم به ترتیب از ادغام میانگین و حداکثری ۱۵ × ۱۵ استفاده کرد که خروجی‌های این مسیرهای کمکی قبل از لایه تماما متصل، با ورود به مسیر اصلی به صورت توام در رده‌بندی و قطعه‌بندی استفاده گردد.

References

[1] Kwon, H. J., Koo, H. I., & Cho, N. I. (2023). Understanding and explaining convolutional neural networks based on inverse approach. *Cognitive Systems Research*, *77*(1), 142-152. https://doi.org/10.1016/j.cogsys.2022.10.009

[2] Kale, A. P., Wahul, R. M., Patange, A. D., Soman, R., & Ostachowicz, W. (2023). Development of Deep Belief Network for Tool Faults Recognition. *Sensors*, *23*(4), 1872. https://doi.org/10.3390/s23041872

[3] Dong, S., Wang, P., & Abbas, K. (2021). A survey on deep learning and its applications. *Computer Science Review*, *40*(1), 100379. https://doi.org/10.1016/j.cosrev.2021.100379

[4] Shah, A., Shah, M., Pandya, A., Sushra, R., Sushra, R., Mehta, M., Patel, K., & Patel, K. (2023). A comprehensive study on skin cancer detection using artificial neural network (ANN) and convolutional neural network (CNN). *Clinical eHealth*, *6*(5), 76-84. https://doi.org/10.1016/j.ceh.2023.08.002

[5] He, P., Wang, L., Cui, Y., Wang, R., & Wu, D. (2023). Unsupervised feature learning based on autoencoder for epileptic seizures prediction. *Applied Intelligence*, *53*(18), 20766-20784. https://doi.org/10.1007/s10489-023-04582-9

[6] Bartlett, P. L., Long, P. M., Lugosi, G., & Tsigler, A. (2020). Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, *117*(48), 30063-30070. https://doi.org/10.1073/pnas.1907378117

[7] Zheng, Q., Zhu, J., Li, Z., Tian, Z., & Li, C. (2023). Comprehensive Multi-view Representation Learning. *Information Fusion*, *89*(7-8), 198-209. https://doi.org/10.1016/j.inffus.2022.08.014

[8] Gale, T., Zaharia, M., Young, C., & Elsen, E. (2020, November 09-19). *Sparse GPU Kernels for Deep Learning* [Conference session]. SC20: International Conference for High Performance Computing, Networking, Storage and Analysis, Atlanta, Georgia, USA. https://doi.org/10.1109/SC41405.2020.00021

[9] Choi, R. Y., Coyner, A. S., Kalpathy-Cramer, J., Chiang, M. F., & Campbell, J. P. (2020). Introduction to Machine Learning, Neural Networks, and Deep Learning. *Translational Vision Science & Technology*, *9*(2), 14-14. https://doi.org/10.1167/tvst.9.2.14

[10] Alizadeh, R., Allen, J. K., & Mistree, F. (2020). Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, *31*(3), 275-298. https://doi.org/10.1007/s00163-020-00336-7

[11] Guo, T-D., Liu, Y., & Han, C-Y. (2023). An Overview of Stochastic Quasi-Newton Methods for Large-Scale Machine Learning. *Journal of the Operations Research Society of China*, *11*(2), 245-275. https://doi.org/10.1007/s40305-023-00453-9

[12] Lin, T., Karimireddy, S. P., Stich, S. U., & Jaggi, M. (2021). Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data . In M. Meila, T. Zhang

(Eds.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 6654-6665). arXiv. https://doi.org/10.48550/arXiv.2102.04761

[13] Elshamy, R., Abu-Elnasr, O., Elhoseny, M., & Elmougy, S. (2023). Improving the efficiency of RMSProp optimizer by utilizing Nestrove in deep learning. *Scientific Reports*, *13*(1), 8814. https://doi.org/10.1038/s41598-023-35663-x

[14] Huang, F., Wu, X., & Hu, Z. (2023, April 25-27). *Adagda: Faster adaptive gradient descent ascent methods for minimax optimization* [Conference session]. The 26th International Conference on Artificial Intelligence and Statistics, Valencia, Spain. https://proceedings.mlr.press/v206/huang23a.html

[15] Huk, M. (2020). Stochastic Optimization of Contextual Neural Networks with RMSprop. In N. T. Nguyen, K. Jearanaitanakij, A. Selamat, B. Trawiński, & S. Chittayasothorn (Eds.), *Intelligent Information and Database Systems* (pp. 343-352). Springer International Publishing. https://doi.org/10.1007/978-3-030-42058-1_29

[16] Reyad, M., Sarhan, A. M., & Arafa, M. (2023). A modified Adam algorithm for deep neural network optimization. *Neural Computing and Applications*, *35*(23), 17095-17112. https://doi.org/10.1007/s00521-023-08568-z

[17] Daoud, M. S., Shehab, M., Al-Mimi, H. M., Abualigah, L., Zitar, R. A., & Shambour, M. K. Y. (2023). Gradient-Based Optimizer (GBO): A Review, Theory, Variants, and Applications. *Archives of Computational Methods in Engineering*, *30*(4), 2431-2449. https://doi.org/10.1007/s11831-022-09872-y

[18] Wang, M., & Wu, L. (2024, May 07-11). *The noise geometry of stochastic gradient descent: A quantitative and analytical characterization* [Conference session]. The Twelfth International Conference on Learning Representations, Vienna, Austria. https://doi.org/10.48550/arXiv.2310.00692

[19] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014, November 03-07). *Caffe: Convolutional Architecture for Fast Feature* [Conference session]. *Embedding*. Proceedings of the 22nd Association for Computing Machinery international conference on Multimedia, Orlando, Florida, USA. https://doi.org/10.1145/2647868.2654889

[20] Ceriotti, M., More, J., & Manolopoulos, D. E. (2014). i-PI: A Python interface for ab initio path integral molecular dynamics simulations. *Computer Physics Communications*, *185*(3), 1019-1026. https://doi.org/10.1016/j.cpc.2013.10.027

[21] Hoseini, F., Shahbahrami, A., & Bayat, P. (2019). AdaptAhead Optimization Algorithm for Learning Deep CNN Applied to MRI Segmentation. *Journal of Digital Imaging*, *32*(1), 105-115. https://doi.org/10.1007/s10278-018-0107-6

[22] Yousef, R., Khan, S., Gupta, G., Albahlal, B. M., Alajlan, S. A., & Ali, A. (2023). Bridged-U-Net-ASPP-EVO and Deep Learning Optimization for Brain Tumor Segmentation. *Diagnostics*, *13*(16), 2633. https://doi.org/10.3390/diagnostics13162633

[23] ZainEldin, H., Gamel, S. A., El-Kenawy, E-S. M., Alharbi, A. H., Khafaga, D. S., Ibrahim, A., & Talaat, F. M. (2023). Brain Tumor Detection and Classification Using Deep Learning and Sine-Cosine Fitness Grey Wolf Optimization. *Bioengineering*, *10*(1), 18. https://doi.org/10.3390/bioengineering10010018

[24] Elmezain, M., Mahmoud, A., Mosa, D. T., & Said, W. (2022). Brain Tumor Segmentation Using Deep Capsule Network and Latent-Dynamic Conditional Random Fields. *Journal of Imaging*, *8*(7), 190. https://doi.org/10.3390/jimaging8070190