Islamic Azad University , Shiraz Branch
نشریه تحلیل مدارها، داده ها و سامانه ها
Journal of Circuits, Data and Systems Analysis
sanad.iau.ir/journal/jcdsa

Designing CNNs with Effective Weights Using Genetic Algorithm for Image Classification

Mojtaba Sajadi[1], Mohammad Bagher Tavakoli*[2], Farbod Setoudeh[3], Amir Hossein Salemi[4]

[1]Department of Electrical Engineering, Arak Branch, Islamic Azad University, Arak, Iran
sajadiarak@gmail.com
[2]Department of Electrical Engineering, Arak Branch, Islamic Azad University, Arak, Iran
mb-tavakoli@iau-arak.ac.ir
[3]Department of Electrical Engineering, Arak University of Technology, Arak, Iran
f.setoudeh@arakut.ac.ir
[4]Department of Electrical Engineering, Arak Branch, Islamic Azad University, Arak, Iran
ah-salemi@iau-arak.ac.ir

Abstract: Convolutional neural networks (CNNs) are the most important branch of deep learning (DL) and have experienced rapid development in recent years. A major challenge in using these networks is their large number of parameters, which result in high computational and time costs in real-world applications. In many cases, this increase in costs is due to the design of deeper networks with more parameters for achieving higher accuracy. The present paper employed evolutionary algorithms (EAs) to introduce a method that can identify the best weights and use them to construct more accurate CNNs, hence eliminating the need for deeper networks**.** At the end of the article, the CNN obtained from the proposed algorithm is compared with the best existing CNNs; which shows that the proposed CNN has increased the classification accuracy, while the number of its parameters is much less, and as a result, it saves computing resources and time.

Keywords: Convolutional neural network, Genetic algorithm, Effective weights, Image Classification.

JCDSA, Vol. 2, No. 1, Spring 2024	Online ISSN: 2981-1295	Journal Homepage: https://sanad.iau.ir/en/Journal/jcdsa
Received: 2023-12-23	Accepted: 2024-05-11	Published: 2024-06-05

CITATION

Sajadi, M., et. al., *"Designing CNNs with Effective Weights Using Genetic Algorithm for Image Classification "*, Journal of Circuits, Data and Systems Analysis (JCDSA), Vol. 2, No. 1, pp. 16-25, 2024.
DOI: 00.00000/0000

* Corresponding author

Extended Abstract

1- Introduction

Convolutional Neural Networks (CNN) is one of the most popular and widely used deep learning techniques, which has made significant progress in most machine learning methods in recent years [1-3]. The most important advantage of these networks compared to previous techniques is the extraction of features from images automatically without the need for the presence of a human observer [1]. This has caused the creation of convolutional neural networks with different architectures, each of which contains millions of parameters in convolutional layers to extract features from images. The large number of these parameters makes learning the network very time-consuming, and it also makes it difficult to hyper-adjust the parameters related to learning because every small change in the network configuration will cause a change in several million parameters, and due to the long learning time, behavior analysis The network becomes a problem [4, 5]. Due to the necessity of multiple training of these networks in real applications, it is necessary to use methods to reduce the number of parameters or design a network with optimal parameters [6, 7].

2- Methodology

Before convolutional neural networks entered the field of competition, the most important obstacle for image classification algorithms was the selection of appropriate features from images. These features should be selected considering the desired application in such a way that they represent the desired entity and are resistant to changes such as rotation, resizing, brightness, color, and occlusion. With the advent of convolution networks by Alex and his colleagues [22], finding these filters is done automatically. In this way, at first, thousands of filters are considered in the form of convolution layers, and during learning, suitable values for these filters are obtained. In other words, at the end of learning, we have filters by which we can extract the features that are effective in learning. Fully connected layers (FC) also perform classification work.

In this article, we have designed a mechanism by which more effective filters can be found and used in network training. By using this mechanism, the accuracy increases significantly and there is no need to increase the depth to achieve higher accuracy.

The output of the proposed algorithm is a convolution network that has the same architecture, but its weights are selected in such a way that it has maximum accuracy. This algorithm consists of the well-known steps of the genetic algorithm, the parts of generating the initial population, evaluating the population, producing offspring, and selecting the new generation will be described in the following subsections. At the stage of initial population production, to build the initial population, first N convolution networks are built with the architecture mentioned in the input of the algorithm and their weights are assigned with random values. Then, based on the input F, a subset of the training database, whose number of members (K) is calculated at the beginning of the algorithm in Table 2, is randomly selected and the desired CNN is trained with it. To calculate the fit of each of the networks (individuals), we use the test dataset, which is defined as the input of the algorithm. This dataset is given to each person and their accuracy in classification constitutes the fit value. In most famous datasets, a subset of data is introduced as the test set, which we use to evaluate people [23, 28]. The pseudo code of this section is given in Table 3.

3- Results and discussion

Although all CNN architectures can be used as the basic architecture in the proposed algorithm, it is better to use architectures that have less diversity in their layers. The existence of several FC layers or the use of layers with different filter sizes reduces our control over the analysis of algorithm behavior. Considering this point, we have used Resnet-32 architecture as the basic architecture. Another reason for choosing this architecture is that there are deeper versions of this architecture with more parameters and the results obtained from the final network can be compared with them. The following results are the result of running the proposed algorithm on a Geforce GTX 1080 Ti GPU card. DenseNet-BC and Resnet-32-GA have relatively similar accuracy, but the number of parameters of Resnet-32-GA is about 66% less. But in the case of VGG-19, although the number of parameters of the proposed algorithm is only 0.01 of the parameters of VGG-19, it has increased the accuracy to 2.31%. Genetic CNN and CNN-GA algorithms are algorithms in which the design of the final network is done automatically - for this reason, the time cost of these algorithms is very high and in the range of several GPU days. However, Resnet-32-GA has a 2.51% increase in accuracy compared to the network produced by Genetic CNN, and it has approximately the same accuracy as CNN-GA.

4- Conclusion

The subject of this article was to present a method to find more effective parameters in deep networks and as a result achieve higher accuracy and reduce time and processing costs, while with this work, the need to design deeper networks is eliminated. In order to complete this work, a method based on genetic algorithm was presented, which significantly improved the accuracy compared to the existing methods. Since most of the filters of convolution layers of deep networks are 3*3 filters, our focus in this article has been to select effective filters with the same dimensions, for this reason, Resnet-32 was chosen for conducting tests, and most of its filters have dimensions of 3 It is *3. But in future work, this algorithm can be extended to select filters with other dimensions and, as a result, the ability to implement on more complex networks.

Islamic Azad University , Shiraz Branch
نشریه تحلیل مدارها، داده ها و سامانه ها
Journal of Circuits, Data and Systems Analysis
sanad.iau.ir/journal/jcdsa

طراحی شبکه عصبی کانولوشن با وزن‌های موثر با استفاده از الگوریتم ژنتیک برای طبقه‌بندی تصاویر

مجتبی سجادی[۱]، محمد باقر توکلی[۲*]، فربد ستوده[۳]، امیر حسین سالمی[۴]

۱- گروه مهندسی برق، واحد اراک، دانشگاه آزاد اسلامی، اراک، ایران (sajadiarak@gmail.com)
۲- گروه مهندسی برق، واحد اراک، دانشگاه آزاد اسلامی، اراک، ایران (mb-tavakoli@iau-arak.ac.ir)
۳- گروه مهندسی برق، دانشگاه صنعتی اراک، اراک، ایران (f.setoudeh@arakut.ac.ir)
۴- گروه مهندسی برق، واحد اراک، دانشگاه آزاد اسلامی، اراک، ایران (ah-salemi@iau-arak.ac.ir)

چکیده: شبکه‌های عصبی کانولوشن مهمترین شاخه یادگیری عمیق هستند و در سال‌های اخیر، توسعه سریعی را تجربه کرده‌اند. یک چالش عمده در استفاده از این شبکه‌ها، تعداد زیاد پارامترهای آن‌هاست که منجر به هزینه‌های محاسباتی و زمانی بالا در برنامه‌های کاربردی دنیای واقعی می‌شود. در بسیاری از موارد، این افزایش هزینه‌ها به دلیل طراحی شبکه‌های عمیق‌تر با پارامترهای بیشتر برای دستیابی به دقت بالاتر است. مقاله حاضر از الگوریتم‌های تکاملی برای معرفی روشی استفاده کرده که می‌تواند بهترین وزن‌ها را شناسایی کرده و از آنها برای ساخت شبکه‌های دقیق‌تر استفاده کند؛ در نتیجه نیاز به شبکه‌های عمیق‌تر را از بین می‌برد. در پایان مقاله، شبکه‌ی به دست آمده از الگوریتم پیشنهادی با بهترین شبکه‌های موجود مقایسه شده است که نشان می‌دهد شبکه‌ی پیشنهادی دقت طبقه‌بندی را افزایش داده است؛ در حالی‌که تعداد پارامترهای آن بسیار کمتر است و در نتیجه، باعث صرفه‌جویی در منابع محاسباتی و زمان می‌شود.

واژه‌های کلیدی: شبکه عصبی کانولوشن ، الگوریتم ژنتیک، وزن‌های موثر، طبقه‌بندی تصاویر

نوع مقاله: پژوهشی
DOI: 00.00000/0000
تاریخ ارسال مقاله: ۱۴۰۲/۱۰/۰۲ تاریخ پذیرش مقاله: ۱۴۰۳/۰۲/۲۲ تاریخ چاپ مقاله: ۱۴۰۳/۰۳/۲۷

۱- مقدمه

شبکه‌های عصبی کانولوشن[۱] یکی از محبوب‌ترین و پراستفاده‌ترین روش‌های یادگیری عمیق می‌باشد که در سال‌های اخیر در اکثر روش‌های یادگیری ماشین پیشرفت‌های چشم‌گیری داشته است [۱-۳]. مهم‌ترین مزیت این شبکه‌ها نسبت به روش‌های پیشین، استخراج ویژگی‌ها از تصاویر به صورت اتوماتیک و بدون نیاز به حضور ناظر انسانی می‌باشد [۱]. همین امر سبب شده که شبکه‌های عصبی کانولوشن با معماری‌های مختلف که هر کدام شامل میلیون‌ها پارامتر در لایه‌های کانولوشن جهت استخراج ویژگی‌ها از تصاویر هستند به وجود آیند. تعداد زیاد این پارامترها سبب می‌شود که یادگیری شبکه، بسیار زمان‌بر شود، همچنین تنظیم هایپر-پارامترهای مربوط به یادگیری را مشکل می‌کند؛ زیرا هر تغییر کوچک در پیکربندی شبکه باعث تغییر در چند میلیون پارامتر خواهد شد و با توجه به زمان زیاد یادگیری، تحلیل رفتار شبکه مشکل می‌شود [۴, ۵].

با توجه به لزوم آموزش چندباره‌ی این شبکه‌ها در کاربردهای واقعی، به کاربردن روش‌هایی جهت کم کردن تعداد پارامترها و یا طراحی شبکه با پارامترهای بهینه ضروری است [۶, ۷]. تاکنون تلاش‌های زیادی در همین جهت صورت گرفته است که در ادامه به برخی از آن‌ها اشاره می‌کنیم. یک ایده، حذف قسمتی از اطلاعات است به گونه‌ای که تا حد ممکن، به دقت شبکه صدمه‌ای وارد نشود. برای مثال، چاکرابورتی در [۸] برای کم‌تر کردن هزینه‌های محاسباتی و فضای ذخیره‌سازی تعدادی از نقشه ویژگی‌های[۲] لایه‌های کانولوشن را به صورت تصادفی حذف کرده‌است. این روش می‌تواند برای مساله‌هایی که پیچیدگی کمتری دارند مانند طبقه‌بندی پایگاه داده‌ی MNIST[3] [۹] مفید باشد. اما در طبقه‌بندی تصاویر پیچیده‌تر (مانند Caifar10[4] و IMAGENET)

[1] Convolutional neural networks (CNNs)
[2] Feature Map
[3] این پایگاه داده حاوی تصاویر اعداد دست‌نویس انگلیسی می‌باشد که با در نظر گرفتن اندازه‌ی کوچک تصاویر، تعداد اعداد و پس‌زمینه‌ی ساده، از پیچیدگی کمتری نسبت به سایر پایگاه داده‌های این حوزه برخوردار است.
[4] این پایگاه داده‌ها حاوی تصاویری از دنیای واقعی هستند که به دلیل وجود زاویه‌های مختلفی از اشیاء، چرخش، رنگ بندی‌های متفاوت و حتی پوشانده شدن قسمتی از اشیاء در تصاویر فرآیند طبقه‌بندی آن‌ها از پیچیدگی‌های بالایی برخوردار است.

دقت را به شدت کاهش می‌دهد؛ زیرا هیچگونه ارزش‌گذاری روی اطلاعات حذف شده صورت نمی‌پذیرد و ممکن است نقشه ویژگی‌های های حذف شده حاوی اطلاعات مهمی برای طبقه‌بندی باشند. ایده‌ی دیگر هرس‌کردن، وزن‌ها است. مهم‌ترین چالش در هرس‌کردن، تشخیص وزن‌های با ارزش در یادگیری می‌باشد. یک راه، استفاده از یک مقدار آستانه و حذف وزن‌های با مقادیر کم‌تر از آن است [۱۰]. در روش‌های کاراتر سعی شده از اطلاعات آماری مانند نرم ۲ و تبدیل مسئله به یک مسئله‌ی بهینه‌سازی توسط سری تیلور[1] یا شیوه‌های مبتنی بر شبکه بیزین[2] برای شناسایی وزن‌های با ارزش و حذف دیگر وزن‌ها استفاده شود [۱۱-۱۳]. اما در کل اندازه‌گیری اهمیت پارامتر در شبکه‌های کانولوشن به دلیل تاثیرات متقابل نرون‌ها بسیار مشکل می‌باشد و ممکن است در ابتدا یک اتصال به دلیل وجود وزن‌های مشابه، کم اهمیت به نظر برسد؛ اما با حذف سایر وزن‌ها وجود آن حیاتی خواهد بود.

مقداردهی اولیه به وزن ها از مواردی است که در دقت شبکه‌ی نهایی می‌تواند بسیار موثر باشد [۱۴]. یک راه برای مقداردهی اولیه به وزن‌ها استفاده از وزن‌های شبکه‌هایی است که قبلا آموزش داده شده‌اند. در این روش که به یادگیری انتقالی[3] موسوم است هم‌خوانی معماری شبکه‌ها و اتخاذ راه‌هایی برای پرهیز از یادگیری بیش از حد، حائز اهمیت است [۱۵]. در دسته‌ی دیگری از پژوهش‌ها تحت عنوان جستجوی معماری عصبی (NAS) سعی می‌شود فرآیند یافتن معماری شبکه‌های عصبی کانولوشنی به صورت اتوماتیک انجام شود. در این دسته از الگوریتم‌ها مانند CNN-GA[16] و ENAS[17] بلوک‌هایی متشکل از چندلایه کانولوشن توسط ناظر طراحی می‌شوند و کار الگوریتم پیدا کردن تعداد این بلوک‌ها و نحوه‌ی اتصال آنها به یکدیگر است؛ به طوری که شبکه‌ی حاصل بهترین دقت طبقه‌بندی را ارائه دهد. مشکل این دسته از الگوریتم‌ها زمان بسیار زیاد اجرا (حدود بیست روز) و نیاز آن‌ها به منابع محاسباتی فراوان می‌باشد.

تمامی کارهای پیشین در این زمینه بر انتخاب لایه‌های کانولوشن و چیدمان آن‌ها تمرکز داشته اند؛ در حالیکه هیچ تلاشی برای انتخاب فیلترهای مناسب درون لایه‌ها صورت نگرفته است. در این مقاله، ما روشی را پیشنهاد می‌کنیم که می‌تواند به طور خودکار وزن‌های با ارزش‌تر را شناسایی کند و از آن‌ها برای ساخت شبکه نهایی استفاده کند. در این روش، ساختار شبکه حفظ شده و هر زمان که لازم باشد همه‌ی وزن‌ها شانس حضور در شبکه‌ی نهایی را خواهند داشت.

۲- پیش زمینه

پیش از آنکه شبکه‌های عصبی کانولوشن به میدان رقابت وارد شوند، مهم‌ترین مانعِ پیشِ رویِ الگوریتم‌های طبقه‌بندی تصویر، انتخاب ویژگی‌های مناسب از تصاویر بود. این ویژگی‌ها می‌بایست با در نظر گرفتن کاربرد مورد نظر، به گونه‌ای انتخاب می‌شدند که هم معرف موجودیت مورد نظر باشند و هم در برابر تغییراتی نظیر چرخش، تغییر اندازه، روشنایی، رنگ و انسداد، مقاوم باشند. نکته‌ی قابل توجه این است که در استخراج اکثر این ویژگی‌ها، از عملیات کانولوشن[4] استفاده می‌شود. برای مثال، می‌توان به استخراجِ ویژگی‌هایی نظیر لبه [۱۸]، انحنا [۱۹]، گوشه [۲۰]، SIFT[21] و... اشاره کرد. در واقع، چالشِ اصلی در این زمینه، یافتنِ فیلترهایی با ضرایبِ مناسب بود تا با اعمالِ آن‌ها بر روی تصاویر، ویژگی‌های مورد نظر استخراج شوند. با ظهور شبکه‌های کانولوشن توسط الکس و همکارانش [۲۲] یافتن این فیلترها به صورت اتوماتیک انجام می‌شود. به این صورت که در ابتدا هزاران فیلتر در قالب لایه‌های کانولوشن در نظر گرفته می‌شوند و در خلال یادگیری مقادیر مناسب برای این فیلترها به دست می‌آید. به عبارت دیگر در پایان یادگیری، ما فیلترهایی داریم که توسط آن‌ها می‌توان ویژگی‌های موثر در یادگیری را استخراج نمود. لایه‌های تماماً متصل نیز کار طبقه‌بندی را انجام می‌دهند. حال سوال اینجاست که چه تعداد فیلتر برای طراحی یک شبکه کانولوشن می‌بایست در نظر گرفته شود تا به دقت مورد نظر برسیم؟ متاسفانه برای این سوال جواب واضحی وجود ندارد و در بسیاری از معماری‌ها به صورت تجربی در مورد انتخاب تعداد لایه‌ها و تعداد فیلترها تصمیم‌گیری می‌شود و درصورت عدم دستیابی به جواب مناسب، شبکه‌های عمیق‌تر با تعداد فیلتر بیشتر آزمایش می‌شود. به همین دلیل است که در بسیاری از معماری‌ها چندین نسخه با عمق‌های مختلف وجود دارد. اما آیا به تمام فیلترهایی که در خلال یادگیری به دست می‌آیند نیاز است؟ و آیا می‌توان میزان تاثیر این فیلترها در دقت خروجی را به دست آورد و از فیلترهای موثرتر در کاربردهای مشابه استفاده نمود؟ ما در این مقاله مکانیزمی طراحی کرده‌ایم که توسط آن می‌توان فیلترهای موثرتر را پیدا کرده و در آموزش شبکه از آن‌ها استفاده کرد. با استفاده از این مکانیزم دقت به میزان قابل ملاحظه‌ای افزایش پیدا می‌کند و احتیاج به افزایش عمق برای دست یابی به دقت بالاتر نیست.

۳- الگوریتم پیشنهادی

در این بخش، ابتدا در زیر بخش ۳-۱ چارچوب کلی الگوریتم را بیان می‌کنیم و سپس در زیر بخش‌های ۳-۲ تا ۳-۵ جزئیات هرگام را شرح خواهیم داد. در هر قسمت علاوه بر آوردن شبه کد، جزئیات هر گام نیز شرح داده شده است.

۳-۱- الگوریتم کلی

چارپوب کلی الگوریتم پیشنهادی در جدول (۱) آمده است. در ورودی الگوریتم می‌بایست معماری مورد نظر را معرفی کنیم. تنظیمات مربوط به الگوریتم ژنتیک - اندازه جمعیت و تعداد نسل‌ها- از دیگر ورودی‌های الگوریتم پیشنهادی هستند. همچنین داده‌های آموزش و تست مورد استفاده را نیز در این قسمت مشخص می‌کنیم. خروجی الگوریتم پیشنهادی یک شبکه‌ی کانولوشن است که همان معماری را دارد اما وزن‌های آن به گونه‌ای انتخاب شده‌اند که دارای دقت ماکزیمم باشد.

[1] Taylor expansion
[2] Bayesian optimization-based method
[3] Transfer learning
[4] convolution

این الگوریتم، از گام‌های شناخته شده‌ی الگوریتم ژنتیک تشکیل شده است که قسمت‌های تولید جمعیت اولیه، ارزیابی جمعیت، تولید فرزندان و انتخاب نسل جدید در زیربخش‌های بعد شرح داده خواهند شد.

جدول (۱): شبه کد مربوط به الگوریتم کلی

P: Population size
M: Maximal generation number
D: Training images dataset
F: Fitness function
Begin
$P_0 \leftarrow$ The initial population generated by the pseudo code of Table2
$t \leftarrow 0$
while $t <$ M ***do***
Fitness evaluation for all individual using **pseudo code of Table 3**
$G_t \leftarrow$ new generation from the selected parent using **pseudo code of Table 4**
$P_{t+1} \leftarrow$ New population selection from $P_t \cup G_t$ **pseudo code of Table 6**
$t \leftarrow t + 1$
end while
Return individual with highest fitness as CNN with best weights

جدول (۲): شبه کد مربوط به الگوریتم تولید جمعیت اولیه

N: Population size
Model: CNN Architecture Model
T: Precision threshold
F: Subset fraction factor

Begin
$M_{1,\dots,N} \leftarrow$ initiate n network based on *Model*
$K \leftarrow$ number of dataset elements × (1/F)
$P_0 \leftarrow \emptyset$
for $i = 1$ *to* N ***do***
$D_i \leftarrow$ Random subset of dataset with K element
Train M_i while satisfying T
$P_0 \leftarrow P_0 \cup M_i$
end for

Return The initialized population as P_0

جدول (۳): شبه کد مربوط به الگوریتم ارزیابی جمعیت

P: Population
Dt: Test dataset for evaluation
Begin
for all individuals in P ***do***
$value \leftarrow$ classification accuracy on Dt
end for
Return The Population P of the individuals with their fitness values

۳-۲- تولید جمعیت اولیه

در این مرحله برای ساختن جمعیت اولیه ابتدا N شبکه‌ی کانولوشن با معماری‌ای که در ورودی الگوریتم آمده است، ساخته شده و وزن‌هایشان با مقادیر تصادفی مقداردهی می‌شوند. سپس بر اساس ورودی F، یک زیرمجموعه از پایگاه داده‌ی آموزش که تعداد اعضای آن (K) در ابتدای الگوریتم جدول (۲) محاسبه می‌شود، به صورت تصادفی انتخاب شده و *CNN* مورد نظر با آن آموزش می بینند. برای پایگاه داده‌هایی مانند *Imagenet* [23] و *Ms coco* [24] که حاوی تعداد زیادی نمونه در هر کلاس هستند مقدار F برابر با N در نظر گرفته می‌شود. اما در پایگاه داده‌هایی که تعداد نمونه‌های هر کلاس کم می‌باشد مانند *Caltech* [25] و *CIFAR100* [26] می‌بایست مقدار F کوچکتر از N در نظر گرفته شود تا داده‌های بیشتری برای آموزش هر شبکه انتخاب شده و از یادگیری بیش از حد[1] پرهیز شود. همچنین با توجه به این که هر یک از شبکه‌ها فقط با قسمتی از پایگاه‌داده آموزش می‌بینند احتمال اتفاق افتادن یادگیری بیش از حد افزایش می‌یابد. بنابراین حتما می‌بایست در آموزش از افزایش داده‌ها[2] استفاده گردد [۲۷]. آموزش تا جایی ادامه می‌یابد که شبکه‌ها همگرا شوند. از آن جا که آموزش شبکه‌ها به صورت مستقل از یکدیگر انجام می‌شود، برای بالا رفتن سرعت می‌توان این کار را به صورت موازی و روی چند کامپیوتر نیز انجام داد. شِبه‌کد مربوط به این بخش در جدول (۲) آمده است.

۳-۳- ارزیابی جمعیت

برای محاسبه‌ی برازش[3] هر یک از شبکه‌ها (افراد) از دیتاست *Test* که به عنوان ورودی الگوریتم تعریف شده است استفاده می‌کنیم. این دیتاست به هر یک از افراد داده شده و میزان دقت آن‌ها در طبقه‌بندی، مقدار برازش را تشکیل می‌دهد. در اکثر دیتاست‌های معروف یک زیر مجموعه از داده‌ها به به عنوان مجموعه تست معرفی شده است که ما از همان دیتاست به منظور ارزشیابی افراد استفاده می‌کنیم [۲۳, ۲۸]. شبه‌کد این قسمت در جدول (۳) آورده شده است.

۳-۴- تولید نسل جدید

جزئیات تولید نسل جدید در الگوریتم جدول (۴) آمده است. در این مقاله برای تولید نسل جدید از استخر فیلتر[4] استفاده خواهیم کرد. در این الگوریتم ابتدا استخر فیلترها از جمعیت کنونی تهیه می‌شوند، به این صورت که k فرد ($k \geq 2$) که هرکدام یک شبکه عصبی کانولوشنی هستند به صورت تصادفی انتخاب شده و به ازای هر لایه کانولوشن موجود در مدل‌ها، یک استخر فیلتر تشکیل می‌شود. برای مثال اگر $k=4$ و معماری مورد استفاده *Resnet-32* باشد از آنجایی که این شبکه دارای ۳۰ لایه کانولوشن است، ۳۰ استخر فیلتر خواهیم داشت و در استخر فیلتر مربوط به لایه اول که دارای ۶۴*۶۴ فیلتر است، ۴*۶۴*۶۴ فیلتر قرار می گیرد. سپس استخرهای ایجاد شده را به فرایند تولید فرزندان

[1] Overfitting
[2] data augmentation
[3] Fitness
[4] Filter pool

(جدول (۵)) ارسال می‌کنیم که خروجی آن تولید یک فرزند جدید می‌باشد. تمرکز ما در الگوریتم پیشنهادی روی انتخاب فیلترها با ابعاد ۳*۳ جهت ساختن لایه‌های کانولوشن می‌باشد. بر حسب این که چه معماری پایه‌ای انتخاب شده باشد، فرزند تولید شده ممکن است دارای لایه‌هایی با انواع دیگر نیز باشد (*shortcut* ،*Fc*،...) که پارامترهای آن ها به صورت تصادفی مقدار دهی شده‌اند. در نتیجه مقدار برازش فرزند تولید شده به شدت کاهش می‌یابد.

برای غلبه بر این مشکل، پس از تولید فرزند جدید آن را به تعداد E، تکرار آموزش می‌دهیم. مقدار E به معماری پایه‌ی انتخاب شده بستگی دارد. برای معماری‌هایی که تنوع لایه‌های آن‌ها کمتر است، عدد کوچکتر و برای معماری‌های پیچیده‌تر، عدد بزرگتری برای E در نظر گرفته می‌شود. در آزمایشات مشاهده شد با انتخاب $E = 1$ برای معماری‌های ساده تر و انتخاب $E \leq 3$ برای معماری‌های پیچیده‌تر دقت به میزان مورد نظر خواهد رسید. در ادامه به شرح فرایند تولید نسل که در جدول (۵) آمده، خواهیم پرداخت. در ابتدای این الگوریتم، یک شبکه با معماری مورد نظر ساخته می‌شود که وزن‌های آن با مقادیر تصادفی مقداردهی شده‌اند. پس از آن که تعداد فیلترهای مورد نظر(N_p) مشخص شد، تابع $RandomSelect$ فراخوانی می‌شود. در این تابع به تعداد N_p ، هر بار یک فیلتر به صورت تصادفی از استخر مرتبط انتخاب می‌شود و در نهایت فیلترهای انتخاب شده به الگوریتم اصلی باز می‌گردند. در خطی که با * مشخص شده است تمام وزن‌های یک لایه کانولوشن که در $SelectedWeights$ قرار دارند در لایه متناظر در مدل خروجی بارگذاری شده و عملیات برای لایه‌های دیگر ادامه خواهد یافت.

و در زیر بخش ۴-۳ نتایج به دست آمده از اجرای الگوریتم پیشنهادی با نتایج کارهای دیگر مقایسه شده است.

جدول (۴): شبه کد مربوط به الگوریتم تولید نسل جدید

```
P: The Population
E: Number of epochs to train new child
L: Number of convolutional layers with dim 3 × 3

Begin
G ← ∅
while |G| < |P| do
    k ← Randomly select a number
                in range [2 , |P| ]
    S ← Randomly select k individual from P
    for l = 1, … , L do
        Pool_l ← ∅
        F ← Aggrigate filters in
                layer l of S elements
        Pool_l ← Pool_l ∪ F
    end for
    newChild ← the returned individual from
                pseudo code of Table 5
    Train newChild for E epochs
    G ← G ∪ newChild
end
  Return The new generation has been produced as G
```

جدول (۵): شبه کد مربوط به فرایند تولید فرزندان

```
L: Number of convolutional layers with dim 3 × 3
Pool: input pool from  pseudo code of Table 4

Begin
C ← initiate a model with desired architecture
for i = 1, … , L do
    SelectedWeights ← ∅
    N_p ← Number of filters in C. layer_i
    FilterCollection ← RandomSelect(Pool_i , N_p)
    SelectedWeights ← SelectedWeights ∪
                      FilterCollection
  * Load SelectedWeights to layer_i of C
end
Return The generated child as C
```

```
function RandomSelect(Pool , N_p)
Begin
    outFilters ← ∅
    for i = 1, … , N_p do
        SelectedFilter ← random select from Pool
        outFilters ← outFilters ∪ SelectedFilter
    end for
Return outFilters
```

جدول (۶): شبه کد مربوط به الگوریتم انتخاب نسل بعد

```
P_t: The Parent population in epoch t
G_t: Population of offspring produced in epoch t
Begin
A ← P_t ∪ G_t
P_{t+1} ← ∅
while |P_{t+1}| < 0.2 × |P_t| do
    p ← individual with the best fitness in A
    P_{t+1} ← P_{t+1} ∪ p
end while
while |P_{t+1}| < |P_t| do
    p_1, p_2 ← Randomly select two indivisual
                    from A
    p ← Select the one with better fitness
                    from {p_1, p_2}
    P_{t+1} ← P_{t+1} ∪ p
end while

Return The next generation population P_{t+1}
```

۳-۵- انتخاب نسل بعد

یک راه برای انتخاب نسل بعدی، انتخاب اعضای با بهترین مقدار برازش می‌باشد [۲۹]. اگرچه این راه باعث همگرا شدن سریع الگوریتم می‌شود، خطر قرار گرفتن در نقطه‌ی بهینه‌ی محلی را تشدید می‌کند. برای غلبه بر این مشکل ما از روش جستجوی باینری استفاده کرده‌ایم [۲۹, ۳۰]. با استفاده از این روش اعضای با برازش پایین‌تر نیز شانس حضور در نسل بعد را دارند. اما این مشکل وجود دارد که ممکن است اعضایی با داشتن بیشترین برازش در نسل بعد حضور نداشته باشند که باعث همگرایی دیرتر می‌شود. برای غلبه بر این مشکل، ما از روشی شبیه به آنچه در [۱۶] گفته شده استفاده می‌کنیم. به این ترتیب که ۲۰ درصد

جمعیت را از اعضای با بیشترین برازش انتخاب می‌کنیم و سپس باقی اعضا با الگوریتم جستجوی باینری از مجموعه‌های P_t و Q_t انتخاب می‌شوند. شبه‌کد این قسمت در جدول (۶) آمده است.

۴- نتایج

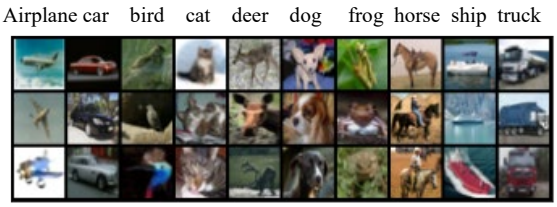
برای ارزیابی کارایی الگوریتم پیشنهادی، تعدادی از الگوریتم‌ها انتخاب شده‌اند که در زیربخش ۴-۱ به آنها اشاره شده‌است. در زیربخش ۴-۲ ساختار دیتاست مورد استفاده برای انجام آزمایشات شرح داده شده است.

۴-۱- رقبا

برای مقایسه کارایی الگوریتم پیشنهادی، شبکه‌های با بیشترین دقت در این حوزه انتخاب شده‌اند که شامل *ResNet*، [31] *DenseNet* و [32] *VGGNet* می‌شوند و همگی جزو برندگان مسابقات *Large-Scale Visual Recognition Challenges* در سال‌های اخیر بوده‌اند [۳۳, ۳۴]. از انواع مختلف *Resnet*، دو مدل *Resnet-110* به علت این که بهترین دقت را در میان مدل‌های *Resnt* دارد و *Resnet-1202* به عنوان پیچیده‌ترین مدل *ResNet* با داشتن بیشترین پارامتر انتخاب شده‌اند [۳۵]. به علاوه از میان انواع مختلف *DenseNet*، مدل *DenseNet-BC* که دارای بهترین دقت در میان مدل‌های *DenseNet* است انتخاب شده است [۳۶]. همچنین دو الگوریتم دیگر که در طراحی آن‌ها از الگوریتم ژنتیک استفاده شده نیز انتخاب شده اند. در *Genetic CNN* یک معماری اولیه توسط طراحان در نظر گرفته می‌شود و از الگوریتم ژنتیک به منظور یافتن جای مناسب لایه‌ها استفاده می‌شود [۳۷]؛ به عبارتی طراحی شبکه به صورت نیمه اتوماتیک انجام می‌شود. اما در *CNN-GA* تمام طراحی شبکه از جمله نوع لایه‌ها و ترتیب قرار گرفتن آن‌ها به صورت اتوماتیک انجام می‌شود [۱۶]. البته باید توجه داشت که هیچ یک از این دو الگوریتم به پارامترهای لایه‌ها توجه ندارند و هدفشان چینش مناسب لایه‌ها در کنار یکدیگر است.

۴-۲- پایگاه داده‌های آزمایش

در این مقاله دو پایگاه داده‌ی *CIFAR10* و *CIFAR100* [۲۶] برای انجام آزمایشات و تعیین دقت طبقه‌بندی انتخاب شده اند. دلیل انتخاب این پایگاه داده‌ها، استفاده‌ی گسترده از آن‌ها به منظور طبقه‌بندی در شبکه‌های کانولوشن می‌باشد و می‌توان الگوریتم پیشنهادی را از لحاظ دقت و کارایی با الگوریتم‌های دیگر مقایسه کرد.



شکل (۱): کلاس‌های داده CIFAR10 به همراه سه نمونه از آنها

پایگاه داده‌ی *CIFAR10* از ۶۰۰۰۰ تصویر با ابعاد ۳۲*۳۲ تشکیل شده که ۵۰۰۰۰ داده به عنوان داده‌های آموزش و ۱۰۰۰۰ داده به عنوان داده‌های تست در نظر گرفته شده است. این پایگاه‌داده از ۱۰ کلاس تشکیل شده که هر کدام حاوی ۵۰۰۰ تصویر از داده‌های آموزش می‌باشند. در شکل (۱) تمام کلاس‌های این پایگاه داده به همراه ۳ نمونه از هر کدام نمایش داده شده‌است. ساختار *CIFAR100* همانند *CIFAR10* می‌باشد با این تفاوت که به جای ۱۰ کلاس از ۱۰۰ کلاس تشکیل شده‌است. بنابراین تعداد داده‌های آموزش برای هر کلاس ۵۰۰ عدد می‌باشد. همان‌گونه که در بخش‌های قبل ذکر شد هنگام استفاده از الگوریتم پیشنهادی، داده‌های آموزش به منظور یادگیری شبکه‌های مختلف به قسمت‌هایی تقسیم می‌شوند و برای پرهیز از یادگیری بیش از حد[1] استفاده از مکانیزم‌های افزایش داده‌ها[2] اجتناب‌ناپذیر است. به همین دلیل، هنگام انتخاب هر تصویر از داده‌های آموزش، یک یا چند مورد از تبدیل‌های چرخش، برش و آینه روی آن اعمال می‌شود [۳۸]. همچنین در آزمایش‌ها، از داده‌های تست دیتاست‌های *CIFAR10* و *CIFAR100* برای محاسبه برازش افراد استفاده شده است.

۴-۳- نتایج تجربی

با اینکه از تمام معماری‌های شبکه عصبی کانولوشنی می‌توان به عنوان معماری پایه در الگوریتم پیشنهادی استفاده کرد، اما بهتر است از معماری‌هایی استفاده شود که تنوع کمتری در لایه‌های خود داشته باشند. وجود چندین لایه تماماً متصل و یا استفاده از لایه‌ها با اندازه فیلترهای متفاوت، تسلط ما را بر تحلیل رفتار الگوریتم کاهش می‌دهد. با در نظر گرفتن این نکته، ما از معماری *Resnet-32* به عنوان معماری پایه استفاده کرده‌ایم. دلیل دیگر انتخاب این معماری، این است که نسخه‌های عمیق‌تر آن با پارامترهای بیشتر وجود دارند و می‌توان نتایج به دست آمده از شبکه نهایی را با آن‌ها مقایسه کرد. نتایجی که در ادامه آمده حاصل از اجرای الگوریتم پیشنهادی روی یک کارت *GPU* مدل *Geforce GTX 1080 Ti* می‌باشد.

در ادامه ابتدا تنظیمات اولیه الگوریتم پیشنهادی و مقداردهی به پارامترها شرح داده خواهد شد. سپس نتایج به دست آمده با کارهای مشابه مقایسه می‌شود. برای تولید کردن جمعیت اولیه (الگوریتم ۲) تعداد جمعیت $p_0 = 10$ و حد آستانه‌ی توقف $T_s = .02$ در نظر گرفته شده است و با توجه به تعداد نمونه‌های هر کلاس $F = 5$ در نظر گرفته شده‌است؛ یعنی هر شبکه با یک پنجم داده‌های آموزش یعنی ۱۰۰۰۰ نمونه آموزش می بینند. پس از آموزش شبکه‌ها، با در نظر گرفتن داده‌های تست به عنوان D_f، مقدار برازش برای هر یک از افراد جمعیت محاسبه می‌شود (الگوریتم ۳). دقت به دست آمده برای این شبکه‌ها حدود ۷۵/۶ ٪ می‌باشد. نکته‌ی قابل توجه این است که هر یک از شبکه‌ها قسمتی از داده‌های تست را درست طبقه بندی می‌کنند و برخی از تصاویر که در یک شبکه اشتباه تشخیص داده شده است، ممکن است در شبکه‌های دیگر درست طبقه بندی شده باشند.

[1] overfitting

[2] data augmentation

با در نظر گرفتن این مطلب و یافتن تصاویری که حداقل در یک شبکه درست طبقه بندی شده باشند در مجموع دقت تشخیص داده‌های تست ۹۷/۶ ٪ به دست آمد. در نتیجه می‌توان گفت تمام فیلترهایی که جهت طبقه‌بندی داده‌های تست با دقت ۹۷/۶ ٪ نیاز هستند، در مجموع فیلترهای این شبکه‌ها وجود دارند و فقط می‌بایست فیلترهای موثر، شناسایی شده و در ساختن شبکه نهایی به کار برده شوند که این امر هدف الگوریتم پیشنهاد شده می‌باشد. در ادامه با استفاده از الگوریتم جدول (۴) نسل جدید ساخته می‌شود. از آنجا که تنوع لایه‌ها در معماری Resnet-32 به عنوان معماری پایه انتخاب شده زیاد نیست و فقط یک لایه‌ی تماماً متصل دارد با در نظر گرفتن $E = 1$ در الگوریتم جدول (۴)، پارامترهای لایه تماماً متصل، تنظیم شده و فرزند تولید شده مقدار برازش مورد نظر را به دست می‌آورد. در نهایت، توسط الگوریتم موجود در جدول (۶)، نسل بعدی از بین جمعیت کنونی و فرزندان تولید شده انتخاب می‌شود و یک چرخه‌ی اجرای الگوریتم به پایان می رسد. اجرای الگوریتم تا رسیدن به دقت مورد نظر در چرخه‌های بعد ادامه می‌یابد و در پایان شبکه نهایی که فرد با بیشترین مقدار برازش در نسل آخر است با نام Resnet-32-GA به عنوان خروجی الگوریتم در نظر گرفته می‌شود.

در جدول (۷)، نتایج به دست آمده از اجرای این شبکه با شبکه‌های دیگر مقایسه شده است. در ستون اول نام رقبا آمده است و ردیف‌های اول و دوم، دقت طبقه‌بندی روی دیتاست‌های CIFAR10 و CIFAR100 را نشان می‌دهند و در ردیف سوم، تعداد پارامترهای هر الگوریتم آورده شده است. بالاخره در ردیف آخر زمان تقریبی اجرای الگوریتم‌ها بر حسب GPU Day با یکدیگر مقایسه شده است. یک GPU Day معادل اجرای الگوریتم روی یک GPU به مدت یک روز است [۳۹]. تمام الگوریتم‌ها به غیر از الگوریتم‌هایی‌که با * مشخص شده‌اند با استفاده از دیتاست‌های مشخص شده آموزش داده شده‌اند و برای مقایسه عادلانه، در آموزش آن‌ها از همان افزایش داده‌هایی که برای الگوریتم پیشنهادی استفاده کرده‌ایم، بهره برده شده‌است. در مورد الگوریتم‌هایی که با * مشخص شده‌اند به دلیل این که زمان آموزش آن‌ها بسیار زیاد بوده و علاوه بر آن، در هر بار اجرای الگوریتم ممکن است شبکه‌ی تولید شده توسط آن‌ها دارای ساختار متفاوتی باشد، به گزارش‌های داده شده در مقالات مرتبطشان بسنده شده است [۱۶, ۳۷].

دقت شبکه نهایی به دست آمده توسط الگوریتم پیشنهادی که با عنوان Resnet-32-GA شناخته می‌شود نسبت به Resnet-32، ۴/۱۳ ٪ افزایش داشته است؛ با اینکه هر دو دارای یک معماری با تعداد پارامترهای یکسان هستند. همچنین دقت Resnet-32-GA نسبت به دیگر شبکه‌های هم خانواده‌ی خود یعنی Resnet-110 و Resnet-1202 به ترتیب ۱/۶۵ ٪ و ۳/۰۶ ٪ افزایش داشته؛ در صورتی که تعداد پارامترهای آن نسبت به Resnet-110 و Resnet-1202 به ترتیب ۸۴ ٪ و ۹۷ ٪ کاهش داشته است. به عبارت دیگر در شبکه Resnet به جای افزایش عمق که باعث افزایش پیچیدگی و افزایش هزینه‌های محاسباتی و زمانی می‌شود، می‌توان با استفاده از الگوریتم پیشنهادی وزن‌های بهینه را انتخاب کرده و به دقت مورد نظر دست یافت. شبکه‌های DenseNet-BC و Resnet-32-GA دارای دقت نسبتاً مشابهی هستند؛ اما تعداد پارامترها Resnet-32-GA حدود ۶۶ ٪ کمتر است. اما در مورد VGG-19، با این که تعداد پارامترهای الگوریتم پیشنهادی تنها ۰/۰۱ پارامترهای VGG-19 است دقت را به مقدار ۲/۳۱ ٪ افزایش داده‌است. الگوریتم‌های Genetic-CNN و CNN-GA الگوریتم‌هایی هستند که طراحی شبکه نهایی آن‌ها به صورت اتوماتیک صورت می‌گیرد؛ به همین دلیل هزینه زمانی این الگوریتم‌ها بسیار بالا و در حد چندین GPU Day است. با این حال Resnet-32-GA نسبت به شبکه تولید شده توسط Genetic CNN، ۲/۵۱ ٪ افزایش دقت داشته و دقتی مشابه CNN-GA دارد.

در جدول (۸) مقایسه بین فاکتورهای زمانی شبکه تولید شده توسط الگوریتم پیشنهادی و Resnet-32 که دارای پارامترهای یکسان هستند صورت گرفته است. همانطور که در جدول دیده می‌شود زمان اجرای الگوریتم پیشنهادی اختلاف چشم‌گیری با اجرای Resnet-32 ندارد؛ در حالیکه دقت را به میزان قابل توجهی افزایش داده‌است. گفتنی‌است ۷۰ ٪ زمان الگوریتم پیشنهادی صرف مرحله‌های مقداردهی اولیه جمعیت[1] و آموزش شبکه‌های میانی (پارامتر E در الگوریتم جدول (۵)) می‌شود که هرکدام از این مرحله‌ها را می‌توان به صورت همزمان روی افراد جمعیت اجرا کرد و در صورت استفاده از یک ساختار برای موازی‌سازی، زمان اجرای الگوریتم به میزان قابل توجهی کاهش خواهد یافت.

۵- نتیجه‌گیری و کارهای آتی

موضوع این مقاله ارائه روشی جهت یافتن پارامترهای موثرتر در شبکه‌های عمیق و در نتیجه دست یافتن به دقت بالاتر و کاهش هزینه‌های زمانی و پردازشی بود؛ ضمن اینکه با این کار، احتیاج به طراحی شبکه‌های عمیق‌تر از بین می‌رود. جهت انجام این کار، روشی بر پایه‌ی الگوریتم ژنتیک ارائه گردید که نسبت به روش‌های موجود، دقت به میزان قابل ملاحظه‌ای بهبود یافت. از آنجایی که اکثر فیلترهای لایه‌های کانولوشن شبکه‌های عمیق را فیلترهای ۳*۳ تشکیل می‌دهند تمرکز ما در این مقاله، انتخاب فیلترهای موثر با همین ابعاد بوده است به همین علت برای انجام تست‌ها *Resnet-32* انتخاب شد که اکثر فیلترهای آن با ابعاد ۳*۳ می‌باشد. اما در کارهای آینده می‌توان این الگوریتم را به انتخاب فیلترهای با ابعاد دیگر و در نتیجه توانایی پیاده‌سازی روی شبکه‌های پیچیده‌تر تعمیم داد. ضمنا الگوریتم پیشنهادی دارای قابلیت موازی‌سازی است که توسط آن می‌توان زمان اجرای الگوریتم را کاهش داد.

[1] population initialization

جدول (۷): مقایسه بین الگوریتم پیشنهادی و رقبای پیشرفته از نظر صحت طبقه بندی

	Manually designed					Automatic designed		Resnet-32-GA
	Resnet-32	**Resnet-110**	**Resnet-1202**	**VGG-19**	**DenseNet-BC**	**Genetic * CNN**	**CNN-GA***	
CIFAR10 (%)	91.28	93.76	92.35	93.10	95.33	92.90	95.22	95.10
CIFAR100 (%)	70.02	74.10	71.86	71.23	77.08	70.97	77.97	76.10
Number of Parameters	0.27M	1.7M	77.08M	20.04M	0.8M	-	2.9M	0.27M
Execution Time	Less than a GPU day	Less than a GPU day	Less than a GPU day	Less than a GPU day	Less than a GPU day	17 GPU day	35 GPU day	Less than a GPU day

جدول (۸): مقایسه زمانی بین دو الگوریتم Resnet-32 و Resnet-32-GA

Resnet-32-GA		Resnet-32	
زمان لازم برای ساختن جمعیت اولیه:	10920 s	زمان هر تکرار:	397.15s
زمان هر تولید نسل:	1110 s	تعداد تکرارها برای همگرا شدن:	60
تعداد نسل ها :	15		
زمان کلی:	**27,570 s**	**زمان کلی:**	**25,814 s**

مراجع

[1] Alzubaidi, L., et al.: Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data 8, 53 (2021). https://doi.org/10.1186/s40537-021-00444-8.

[2] Dhillon, A., Verma, G.K.: Convolutional neural network: a review of models, methodologies and applications to object detection. Prog Artif Intell 9, 85–112 (2020). https://doi.org/10.1007/s13748-019-00203-0.

[3] Yao, G., Lei, T., Zhong, J.: A review of convolutional-neural-network-based action recognition. Pattern Recognition Letters, 118, 14-22 (2019). https://doi.org/10.1016/j.patrec.2018.05.018.

[4] Andonie, R.: Hyperparameter optimization in learning systems. Journal of Membrane Computing, 1(4), 279–291 (2019). https://doi.org/10.1007/s41965-019-00023-0

[5] Khalid, R., Javaid, N.: A survey on hyperparameters optimization algorithms of forecasting models in smart grid. Sustainable Cities and Society, 61, 102275 (2020). https://doi.org/10.1016/j.egyr.2022.09.188

[6] Pietron, M., Wielgosz, M.: Retrain or not retrain?-Efficient pruning methods of deep CNN networks. In Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part III 20. Springer International Publishing, 452-463 (2020). http://dx.doi.org/10.1007/978-3-030-50420-5_34.

[7] Stojanovic, V., et al.: A service-oriented approach for classifying 3D points clouds by example of office furniture classification. In Proceedings of the 23rd International ACM Conference on 3D Web Technology.1-9 (2018). https://doi.org/10.1145/3208806.3208810.

[8] Chakraborty, S., et al.: Feature map reduction in cnn for handwritten digit recognition. In Recent Developments in Machine Learning and Data Analytics. Springer, 143-148 (2019). http://dx.doi.org/10.1007/978-981-13-1280-9_14.

[9] LeCun, Y., The MNIST database of handwritten digits. http://yann. lecun. com/exdb/mnist/, (1998).

[10] Han, S., H. Mao., Dally, W. J.: Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149, (2015).

[11] Choudhary, T., et al.: Inference-aware convolutional neural network pruning. Future Generation Computer Systems, 135, 44-56 (2022). https://doi.org/10.1016/j.future.2022.04.031.

[12] Molchanov, P., et al.: Pruning convolutional neural networks for resource efficient inference. arXiv preprint arXiv:1611.06440, (2016).

[13] Guo, Y., Yao, A., Chen, Y.: Dynamic network surgery for efficient dnns. Advances in neural information processing systems, 29 (2016). https://doi.org/10.48550/arXiv.1608.04493.

[14] Narkhede, M.V., P.P. Bartakke, and M.S. Sutaone, A review on weight initialization strategies for neural networks. Artificial intelligence review, 2022. 55(1): p. 291-322. https://arxiv.org/abs/2310.08109

[15] Xu, Z., et al., Initializing Models with Larger Ones. arXiv preprint arXiv:2311.18823, 2023. https://arxiv.org/abs/2404.01383.

[16] Sun, Y., et al., Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE transactions on cybernetics, 2020. 50(9): p. 3840-3854.

[17] Xie, Y., et al., Automated design of CNN architecture based on efficient evolutionary search. Neurocomputing, 2022. 491: p. 160-171.

[18] Ganesan, P., Sajiv, G.| A comprehensive study of edge detection for image processing applications. In 2017 international conference on innovations in information, embedded and communication systems (ICIIECS), IEEE, 1-6 (2017). https://doi.org/10.1109/ICIIECS.2017.8275968.

[19] Tang, Y., et al.: Principal curvature measures estimation and application to 3D face recognition. Journal of Mathematical Imaging and Vision, 59(2), 211-233 (2017). https://doi.org/10.1007/s10851-017-0728-2.

[20] Possa, P.R., et al.: A multi-resolution FPGA-based architecture for real-time edge and corner detection. IEEE

Transactions on Computers, 63(10), 2376-2388 (2013). http://doi.org/10.1109/TC.2013.130.
[21] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision, 60(2), 91-110 (2004). https://doi.org/10.1023/B:VISI.0000029664.99615.94.
[22] Krizhevsky, A., Sutskever, I., Hinton, G. E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM, 60(6), 84-90 (2017). http://dx.doi.org/10.1145/3065386.
[23] Russakovsky, O., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision, 115(3), 211-252 (2015). http://dx.doi.org/10.1007/s11263-015-0816-y.
[24] Lin, T. Y., et al.: Microsoft coco: Common objects in context. In European conference on computer vision, Springer, 740-755 (2014). https://doi.org/10.1007/978-3-319-10602-1_48.
[25] Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset, (2007).
[26] Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. (2009).
[27] Perez, L., Wang, J.: The effectiveness of data augmentation in image classification using deep learning. ArXiv preprint arXiv:1712.04621, (2017).
[28] Everingham, M., et al.: The pascal visual object classes (voc) challenge. International journal of computer vision, 88(2), 303-338 (2010). https://doi.org/10.1007/s11263-009-0275-4.
[29] Miller, B. L., Goldberg, D. E.: Genetic algorithms, tournament selection, and the effects of noise. Complex systems, 9(3), 193-212 (1995). https://doi.org/10.1162/evco.1996.4.2.113.
[30] Blickle, T.: Tournament selection. Evolutionary computation, 1,181-186 (2000). http://dx.doi.org/10.1007/978-3-642-16493-4_19.
[31] Sun, Y., et al.: Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE transactions on cybernetics, 50(9), 3840-3854 (2020). http://dx.doi.org/10.1109/TCYB.2020.2983860.
[32] Huang, G., et al.: Densely connected convolutional networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, 4700-4708 (2017). http://dx.doi.org/10.1109/CVPR.2017.243.
[33] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ArXiv preprint arXiv:1409.1556, (2014).
[34] Singh, R.V.: ImageNet Winning CNN Architectures–A Review. Rajat Vikram Singh–Institue of Software Research at Carnegie Mellon University, (2015).
[35] Kumar, N., Kaur, N., Gupta, D.: Major convolutional neural networks in image classification: a survey. In Proceedings of International Conference on IoT Inclusive Life (ICIIL 2019), NITTTR Chandigarh, India, Springer, 243-258 (2020). http://dx.doi.org/10.1007/978-981-15-3020-3_23.
[36] He, K., et al.: Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770-778 (2016). http://dx.doi.org/10.1109/CVPR.2016.90.
[37] Chen, L., et al.: Review of image classification algorithms based on convolutional neural networks. Remote Sensing, 13(22), 4712 (2021). https://doi.org/10.3390/rs13224712.
[38] Xie, L., Yuille, A.: Genetic cnn. In Proceedings of the IEEE international conference on computer vision, (2017).
[39] Shorten, C., Khoshgoftaar, T. T.: A survey on image data augmentation for deep learning. Journal of big data, 6(1), 1-48 (2019). https://doi.org/10.1186/s40537-019-0197-0.
[40] Sun, Y., et al.: Evolving deep convolutional neural networks for image classification. IEEE Transactions on Evolutionary Computation, 24(2), 394-407 (2019). https://doi.org/10.48550/arXiv.1710.10741.