Squeeze and Excitation Block Based Neural Architecture Search with Randomization for CNN Construction

Katia Benali
LCSI, Ecole nationale Supérieure d'Informatique (ESI), Oued Smar
Algiers, Algeria
hk_benali@esi.dz

Lydia Bouzar-Benlabiod
Jodrey School of Computer Science, Acadia University, Wolfville, NS, Canada
LCSI, Ecole nationale Supérieure d'Informatique (ESI), Oued Smar
Algiers, Algeria
lydia.bouzar-benlabiod@acadiau.ca

Andrew McIntyre
Jodrey School of Computer Science, Acadia University, Wolfville, NS, Canada
andrew.mcintyre@acadiau.ca

***Abstract*—Neural Architecture Search (NAS) is a method for automatically designing neural networks. In this paper, we propose a block based NAS method that implements Randomization based transformation to generate the search space. The objective is to maximize the performance of a Convolutional Neural Network (CNN) model to detect the presence of tumors in mammography images, while minimizing the model size and complexity. The performance of two variants of our solution are compared. The first is based on Inception blocks and the second on a Squeeze and Excitation Inception block. The approaches are tested on the standard MNIST dataset as well as on the KAU-BCMD mammogram dataset. 99.19 % and 98.43 % accuracy were achieved using the two datasets, respectively.**

***Index Terms*—Deep learning, Neural networks design, Neural networks optimization, Neural Architecture Search, Breast cancer**

I. Introduction

Deep Learning (DL) is a branch of Artificial Intelligence (AI) that aims to teach machines to do specific tasks. The intricate and computationally intensive nature of these algorithms means a significant need for computing power and storage, mainly due to the large number of neural network layers and parameters needed to achieve precise results. Consequently, researchers thought of designing less complex models by proposing theoretical bounds to the neurons and number of layers [1], as well as through constructive [2] and pruning [3] algorithms. Nonetheless, the predominant approach for designing DL models is the empirical experimentation method [4] demanding expertise and time investment for fine-tuning the model hyperparameters. Several studies are attempting to automate the entire design process of DL models, using optimization algorithms [5] and Neural Architecture Search (NAS) [6] techniques to reduce the architectural complexity and maximize the model accuracy.

This paper proposes a NAS-based technique to construct an optimized CNN model for mammogram classification. A new NAS approach is proposed to automatically generate the CNN architecture given a specific dataset. The solution starts with a single block and applies Randomization-based transformations to create the search space. Each model of the search space is partially trained and two models, that meet predefined criteria, are selected and then are used to generate other models of the search space. In this work we focus on one mammogram dataset: KAU-BCMD [7].

This paper is structured as follows: section 2 is dedicated to list the main related works in the field of NAS specifically applied in computer vision. Section 3 summarises the proposed approach and details its different modules, section 4 illustrates the results of testing our solution on a general purpose image dataset MNIST and on a mammogram dataset.

II. Background

The determination of an optimal neural network architecture for a given problem remains a major challenge and a considerable volume of literature has been devoted to the topic [8]. Regardless of approach, NAS requires three components: search space (representation), search strategy, and performance estimation [9]. Researchers have been investigating these areas with the common goal of exploring the space of architectural variants and exploiting the best regions to find optimal candidate architectures with respect to performance criteria. The most commonly used search strategies include Random Search, Bayesian Optimization, Evolutionary methods, Reinforcement Learning, and Gradient-based methods [9]. The use of Bayesian optimization in the work of Mendoza et al. [10] allowed for the creation of a model that surpassed human experts while RL search has been demonstrated in the works of [11]. Evolutionary NAS (so-called ENAS) methods have recently shown significant promise as a heuristic that employs populations of architecture candidates that are evolved using a Neo-Darwinian metaphor [12].

Modern medical image classification systems largely employ CNN models and several works on this type of neural network have been carried out since the late 1970s [13]. The

2835-5776/24/$31.00 ©2024 IEEE
DOI 10.1109/IRI62200.2024.00022

use of CNNs gained momentum thanks to the significant contribution of Krizhevsky et al. [14]. The use of CNNs expanded; further progress was made using deeper architectures [15]. Manually designed and parameterized solutions include [16], who proposed a CNN model dedicated to the segmentation of medical images, specifically mammograms, and trained on the DDSM dataset. [17], on the other hand, proposed a Hybrid CADx (DBN + CNN), where a cascade of deep learning classifiers and random forests was used to detect masses in mammograms, using the DDSM and INbreast datasets. [18] also proposed a CADx dedicated to the detection of cancerous masses from mammograms. CADx is composed of CNNs, and due to insufficient training images to train a deep CNN model, transfer learning was employed to train CNN models for mass detection in mammographic images. Lévy et al. proposed several models for the classification of breast masses: Baseline, AlexNet, and GoogleNet [19]. Despite being trained on the same dataset (DDSM), the three models obtained heterogeneous results and different accuracies. Salama et al. [20] proposed a new framework for the segmentation and classification of breast cancer images. Their work involves training a U-Net model to segment the breast area from mammography images and applying a number of modern architectures to classify mammograms from the MIAS, DDSM, and a subset of the DDSM dataset (CBIS-DDSM) into benign and malignant. The InceptionV3 model returned the best result, particularly with the DDSM dataset, with an accuracy of 98.87 %.

III. Methodology

This section details our proposed NAS method for designing an optimal CNN architecture given a specific dataset. Our solution follows the schema provided in figure 1. It is a pipeline of modules that collaborate to create an optimal CNN model dedicated to image classification. The pipeline takes an image dataset as input and outputs an optimal CNN. The pipeline is composed of three modules: *(1) The Search Space Generation (SSG) module*: determines the neural network architectures that the NAS method can discover. This step involves constructing CNNs that constitute the space to explore. Our solution's search space generation is based on a combination of operations on blocks and layers. Architectures are generated through transformations (using the concept of Randomization [21]). *(2) Model Training (MT) module*: This step involves training the models generated by the previous step. Models are partially trained. After training the generated models, their performance is assessed. *(3) Model Selection (MS) module*: The goal of this step is to select the best models to pass to the next iteration. The selection is based on two criteria: accuracy and, model size and complexity represented by the number of parameters.

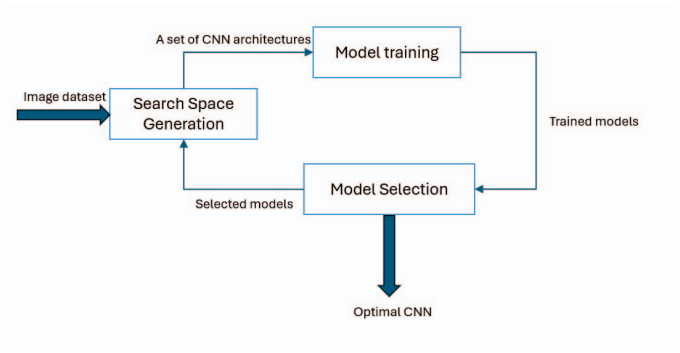

Fig. 1. Overall diagram of the solution.

A. The Search Space Generation module

This module uses a structured search space made up of models that can be built given a specific block. We define two NAS approaches each method bases the search space construction on a specific CNN block:

1) **The Inception block with dimension reduction**: used in GoogleNet architecture [22], the main advantage of the Inception block with dimension reduction is the reduction in computation cost by introducing 1x1 convolution layers to the naïve Inception block architecture (figure 2) in order to limit the number of input channels. This approach is called Block-based NAS BbNAS.

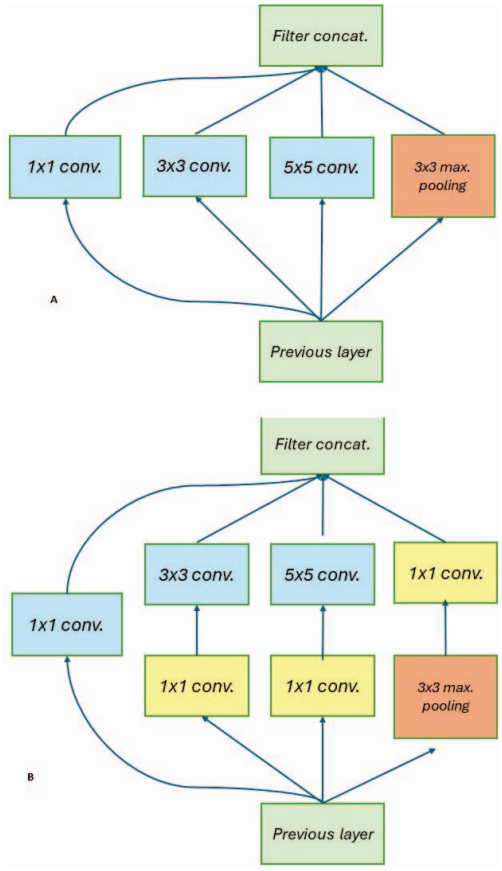

Fig. 2. Architecture of the inception block : Naïve version (a) and with dimension reductions (b) [22].

2) **The Squeeze and Excitation (SE) block** [23]: The main idea behind the SE block is to improve channel interdependencies with minimal impact on the computational cost. The SE-inception block's architecture is shown in figure 3. The inception block is followed by a Squeeze and Excitation branch. The first layer, the global pooling, is used to squeeze global spatial information into a channel descriptor. Then the excitation operation captures channel-wise dependencies, and is achieved by the rest of the layers forming a gating mechanism composed of two fully-connected (FC) separated by a ReLU layer and followed by a sigmoid activation. The final output of the block is obtained by a channel-wise multiplication between the output of the Squeeze and Excitation branch and the feature map. We call this approach SE-BbNAS.

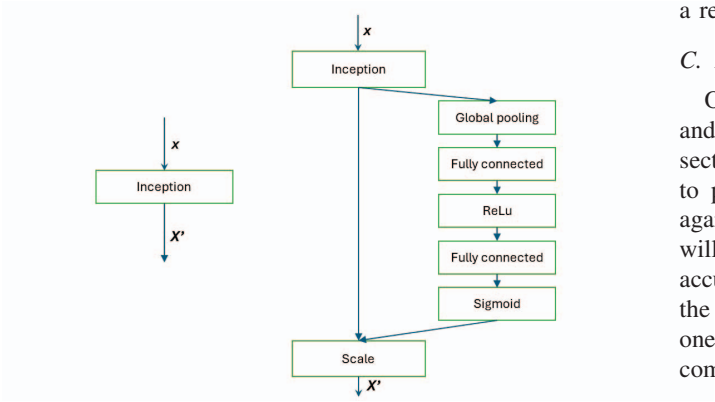

Fig. 3. Architecture of the Squeeze and Excitation inception block [22].

In addition to block based operations, the two approaches manipulate the model layers.
The search space is gradually built up using Randomization [21]. The concept of Randomization has been defined as a technique for generating new knowledge from previously validated one through transformations. In this work, Randomization principle will be used to generate "children" models. The four transformations that are implemented in the SSG module are the following:

1) $add\_layer(G, t)$ operation: adds a type t layer to the graph G. In CNN context, the layer to add is a convolutional layer followed by a pooling layer.
2) $add\_block(G, b)$ operation: This operation adds a block of type b to the graph G. The block to add is either an Inception block or an Inception and Squeeze and Excitation block followed by a fully connected layer.
3) $widen(G, u, n)$ operation: widens a layer u by n. It adds n filters to a convolutional layer. Note that in the case where we want to expand a convolutional layer belonging to an inception block and to maintain the decreasing aspect of the number of convolutions, we increase the first four layers of the block.
4) $concat(G, u)$ operation: concatenates layer u and $u-1$.

These transformations are first run on the initial model to generate children models, and then run on selected models at each iteration. A single iteration generates 8 architectures.

B. Model Training module

Once the models have been generated through randomization, and in order to proceed to the next iteration, it is necessary to determine the smallest model and the best model among them. The smallest model is the one with smallest number of parameters. To determine the best model, we need to calculate the loss and accuracy of every generated model. Thus, the models need to be trained. For training the models, we consider a learning rate of $lr = 0.001$. We use the L2 regularization method with weight decay $wd = 0.02$. Early stopping is also implemented, it stops the training as soon as the chosen metric remains constant or increases. This allows a reduction in training time.

C. Model Selection module

Once the models have been generated using Randomization and trained according to the process described in the previous section, we need to select the best model and smallest model to pass on to the next iteration, to run the transformations again. At the end of the generation process, our search space will consist of eight models. The training result assigns an accuracy for each model. The best model is then selected on the basis of model accuracy, and the smallest model is the one with the fewest parameters. As the search space is not complex, we use an exhaustive search to browse it.

D. Overall view of the method

Figure 4 illustrates how our solution works. Each node represents an intermediate CNN model, and is designated by a pair $(x - y)$ which states for "The y model from iteration x". Our solution starts with an initial model (at the head of the diagram), then through transformations , "children" models are generated and then trained. We consider the initial architecture as a basic CNN:
$Initial\ model = Convolution\ layer \rightarrow Pooling\ layer \rightarrow Designated\ block \rightarrow Pooling\ layer \rightarrow Flatten\ layer \rightarrow Fully\ \ Connected\ \ layer$ At each iteration, the selected models are passed on to the next iteration to undergo the same transformations in order to generate new "children" models, which will also be trained and evaluated. The process thus continues with transformation and selection until a stopping criterion is reached. In our case, it is when the accuracy begins to decline; i.e., n^{th} best model's accuracy is less than $n-1^{th}$ best model's accuracy. The output is then the last best model generated, which will be considered the optimal CNN for the input dataset.

IV. Tests and results

In this section, an overview of the datasets is provided followed by a description of the utilized metrics. This section reports on results of testing every module of the proposed solution separately and the final results when testing the

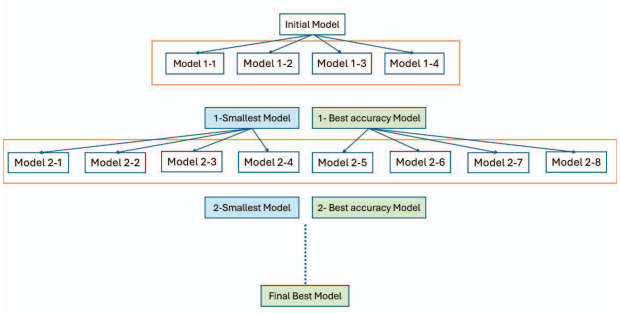

Fig. 4. Comprehensive overview of the method's operation.

proposed solution as a whole on two different datasets. The proposed approaches have been tested on Google Colab using an NVIDIA Tesla K80 GPU. The running time was limited to 12 hours.

A. Used dataset description

- *Modified National Institute of Standards and Technology (MNIST) dataset [24]*: It is a dataset of handwritten number images. It contains 60,000 images in the training set and 10,000 images in the test set.
- *King Abdulaziz University Breast Cancer Mammogram Dataset (KAU-BCMD) [7]*: This dataset includes 1416 cases, each case has two images representing the medio-lateral oblique (MLO) view and cranial caudal (CC) view for right and left breast. The dataset includes 5662 images in total. The dataset is divided into 5 classes following the Breast Imaging Reporting and Data System (BI-RADS) classification: class 1: negative, class 2: benign, class 3: probably benign, class 4: suspicious for malignancy, class 5: highly suggestive of malignancy.

B. Used Metrics

In the following a list of the used metrics is provided. Note that FP stands for False Positive, TP for True Positive, FN for False Negative and TN for True Negative.

Accuracy: This metric measures the proportion of correctly predicted instances out of the total number of instances. It determines how close the predicted values are to the actual or true values.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

Parameters number: This is the number of weights used during training. it gives us a glimpse of the size and complexity of the model. Only the number of parameters of the final generated model is reported.

In addition to the previously defined metrics, the execution time, the time that takes the approach to generate the "optimal" architecture, and the loss metrics were used to assess the approach.

C. Notation

In this section the following notation is used:

- $C(a, b)$: convolution layer with a filters of size b x b.
- $P(a, b)$: pooling layer with step a and size b.
- $Concat()$: a layer composed of a concatenation of the two precedent layers.
- $F()$: a flattening layer
- $FC(a)$: fully connected layer, a being the outputs number.
- $IB(a, b, c, d, e, f)$: Inception block with the same parameter values as in [22], i.e., $IB(32, 64, 128, 16, 32, 32)$.
- $SEIB(a, b, c, d, e, f)$: Inception block followed by a Squeeze and Excitation block.

D. Testing the modules separately

The proposed model is composed of three modules: (1) Search Space Generation (SSG) module, (2) Model Training (MT) module and (3) Model Selection (MS) module. The first step is to test each module separately, then the whole solution pipeline will be tested.

1) The Search Space Generation (SSG) module: At iteration i, this module takes in a model and generates the search space. It implements Randomization and performs four transformations: (1) add layer, (2) add a block, (3) widen a layer and, (4) concatenate. The transformations are detailed in section III-A

Let's consider the following input model M: *M =C(64,3) ; P(1,1) ; IB (32,64,128,16,32,32);P(1,1) ; F () ; FC (10)*

SSG module applies the four transformations to the input model and generates four children models as detailed in table I.

TABLE I
SSG MODULE OUTPUT MODELS

Model	Applied transformation	Output model structure
M_0	Add_Layer(M)	C(64,3) P(1,1) IB(32,64,128,16,32,32) P(1,1) C(64,3) P(1,1) F() FC(10)
M_1	Add_Block(M)	C(64,3) P(1,1) IB(32,64,128,16,32,32) IB(32,64,128,16,32,32) P(1,1) F() FC(10)
M_2	Widen(M, 3, 32)	C(64,3) P(1,1) IB(64,64,160,16,64,64) P(1,1) F() FC(10)
M_3	Concat(M, -4)	C(64,3) P(1,1) IB(32,64,128,16,32,32) P(1,1) Concat(l(-4),l(-5)) F() FC(10)

2) The Model Training (MT) module: Once the models are generated, next comes the training step. Each generated model is trained. The model weights are calculated and the accuracy assessed.

Adaptive Moment Estimation (ADAM) optimization algorithm is used while training the models. The used parameters are as follows: *Alpha (learning rate)* : 0.001. *Beta1* : 0.9. *Beta2* : 0.999. *Epsilon* : 10^{-7}. We implement early stopping to stop the model training when the loss function remains constant or increases. The *patience* parameter is set to 2 and, the max epoch number is *10*. The L2 regularisation is used with *Weight_decay = 0.02*.

We use the previously set parameters to train *M*: *M = C(64,3) ; P(1,1) ; IB (32,64,128,16,32,32) ; P(1,1) ; F () ; FC (10)*

MNIST dataset is used with a batch size of 64. Obtained results are summarized in Table II.

TABLE II
MODEL TRAINING MODULE RESULTS ON MNIST DATASET

Training time (s)	Epoch number	Accuracy	Loss
161,07	7	98.61 %	0.069

We notice that the training time is relatively small: 2.7 minutes, an accuracy of 98.61 % has been reached. We also note that the validation accuracy has not significantly increased throughout the epochs, but the loss function has decreased. We notice that the number of epochs is 6 instead of 10, which means that the training was stopped early in order to avoid overfitting and an increase of the loss function.

The MT module has been also tested with the models that were generated by the SSG module (table I).

The results are summarized in table III.

TABLE III
RESULTS OF THE TM USING SSG MODELS WITH MNIST DATASET

Model ID	Training time (s)	Epoch number	Accuracy	Loss	param. number
M_0	232.03	10	98.43 %	0.0851	543738
M_1	185.22	5	98.65 %	0.0708	1730026
M_2	127.22	5	98.42 %	0.0831	2104858
M_3	102.49	4	98.68 %	0.0741	3125178

The next step is to select models for the next iteration.

3) Model Selection (MS) module: This module designates the models that will be passed out to the next iteration. Two criteria are used to select these models: (1) the model size and, (2) the model accuracy.

Based on the results given in table III, M_0 model will be selected, as it has the smallest number of parameters. The M_1 model will also be selected as it represents the best model in terms of accuracy.

E. Testing BbNAS and SE-BbNAS

The proposed NAS architectures are first run with a general dataset: MNIST then tested on KAU-BCMD dataset.

1) Results for MNIST Dataset: Table IV shows the results of using BbNAS and SE-BbNAS on MNIST.

For the MNIST dataset we notice that when using SE-BbNAS approach the returned model records a better accuracy as compared to the one returned by BbNAS. Although the first model's architecture is more complex in terms of the number of parameters and it takes more time to execute.

F. Results for KAU-BCMD dataset

KAU-BCMD dataset is divided into three sets: (1) a training set: 85 % of the original dataset, (2) a test set: 10 % of the dataset and, (3) a validation set: 5 % of the dataset.

TABLE IV
RESULTS OF USING BBNAS AND SE-BBNAS ON MNIST

Used NAS	Returned model	Exec. time	Acc. %	Param. number
BbNAS	C(64,3) P(1,1) IB(32,64,128,16,32,32) P(1,1) C(64,3) C(64,3) P(1,1) F() FC(10)	0.83 h	98.60	572,474
SE-BbNAS	C(64,3) SEIB(32,64,128, 16,32,32) P(1,1) SEIB(32,64,128, 16,32,32) SEIB(32,64,128,16,32,32) SEIB(32,64,128,16,32,32) P(1,1) C(64,1) SEIB(32,64,128 ,16,32,32) P(1,1) F() FC(10)	2.48 h	99.19	2,342,282

Table V summarizes the results of using BbNAS and SE-BbNAS. The returned architecture for BbNAS is composed of:

C(64,3) P(1,1) IB(32,64,128,16,32,32) P(1,1) Concat() C(64,3) P(1,1) Concat() Concat() F() FC(3).

The returned CNN architecture when using SE-BbNAS is:

C(64,3) P(1,1) SEIB(32,64,128,16,32,32) P(1,1) C(64,3) P(1,1) SEIB(32,64,128,16,32,32) P(1,1) F() FC(3).

TABLE V
TESTING BBNAS AND SE-BBNAS ON KAU-BCMD DATASET

Used NAS	Exec. time	Epochs number	Accuracy	Loss	Parameters number
BbNAS	12h	4	84.25 %	0.0851	543,738
SE-BbNAS	12h	2	98.43 %	0.31	2,428,403

To assess the impact of the SE-inception block, we replace it with a regular inception block in the previously returned solution to get the following model:

C(64,3) P(1,1) IB(32,64,128,16,32,32) P(1,1) C(64,3) P(1,1) IB(32,64,128,16,32,32) P(1,1) F() FC(3).

This model is used with the KAU-BCMD dataset. Table VI summarizes the results.

TABLE VI
TESTING THE OBTAINED SE-BBNAS ARCHITECTURE WHEN REPLACING THE SE INCEPTION BLOCK WITH A REGULAR INCEPTION BLOCK.

Dataset	Accuracy	Loss	Parameters number
KAU-BCMD	82.67 %	0.35	2,741,027

The two NAS solutions have been run on the KAU-BCMD dataset which has 5662 images. SE-BbNAS returns a better accuracy, + 14 % compared to BbNAS. We notice that the returned solution is not too complex; it is composed of two convolution layers and two SEIB. When replacing in the

previous model the SEIB by regular inception blocks, the accuracy decreases by 16 %. When running the NAS solutions on KAU-BCMD dataset we reach the time limit available on Google Colab. which is 12 hours. It would be possible to reach a better accuracy if the solution had run for more time.

V. Conclusion

This work proposes two different NAS approaches to automatically build an optimal CNN architecture given a specific dataset. BbNAS and SE-BbNAS use Randomization based transformations to generate the search space. BbNAS is based on Inception blocks which have given good results in state-of-the-art work; the SE-BbNAS approach implements the concept of Squeeze and Excitation SE through the use of SE Inception blocks. Four transformations were implemented.

The approaches were tested on two image datasets: MNIST which contains general images and a mammogram dataset (KAU-BCMD) which contains 5662 mammogram images. The results were promising for the datasets with more images, we reached an accuracy of 99.19 % and 98.43 % when using SE-BbNAS on MNIST and KAU-BCMD respectively. To highlight the importance of the SE block we replaced the SE Inception block with a regular Inception Block in the returned architecture and the accuracy decreased from 98.43 % to 82.67 %.

In future works, we are willing to test the architecture with different blocks such as the transformer attention block that may enhance the results.

References

[1] Eshan Pandey and Santosh Kumar. Bounds on learnability of neural networks. In *Advances in Data and Information Sciences: Proceedings of ICDIS 2022*, pages 109–116. Springer, 2022.

[2] Sudhir Kumar Sharma and Pravin Chandra. Constructive neural networks: A review. *International journal of engineering science and technology*, 2(12):7847–7855, 2010.

[3] M Augasta and Thangairulappan Kathirvalavakumar. Pruning algorithms of neural networks—a comparative study. *Open Computer Science*, 3(3):105–115, 2013.

[4] Sean C Smithson, Guang Yang, Warren J Gross, and Brett H Meyer. Neural networks designing neural networks: multi-objective hyper-parameter optimization. In *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2016.

[5] Maher GM Abdolrasol, SM Suhail Hussain, Taha Selim Ustun, Mahidur R Sarker, Mahammad A Hannan, Ramizi Mohamed, Jamal Abd Ali, Saad Mekhilef, and Abdalrhman Milad. Artificial neural networks based optimization techniques: A review. *Electronics*, 10(21):2689, 2021.

[6] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE transactions on neural networks and learning systems*, 2021.

[7] Asmaa S Alsolami, Wafaa Shalash, Wafaa Alsaggaf, Sawsan Ashoor, Haneen Refaat, and Mohammed Elmogy. king abdulaziz university breast cancer mammogram dataset (kau-bcmd). *Data*, 6(11):111, 2021.

[8] Colin White, Mahmoud Safari, Rhea Sukthanker, Binxin Ru, Thomas Elsken, Arber Zela, Debadeepta Dey, and Frank Hutter. Neural architecture search: Insights from 1000 papers, 2023.

[9] Thomas Elsken, Jan Metzen, and Frank Hutter. Neural architecture search: A survey. 08 2018.

[10] Hector Mendoza, Aaron Klein, Matthias Feurer, Jost Tobias Springenberg, Matthias Urban, Michael Burkart, Maximilian Dippel, Marius Lindauer, and Frank Hutter. *Towards Automatically-Tuned Deep Neural Networks*, pages 135–149. Springer International Publishing, Cham, 2019.

[11] Bowen Baker, Otkrist Gupta, Nikhil Naik, and Ramesh Raskar. Designing neural network architectures using reinforcement learning. *CoRR*, abs/1611.02167, 2016.

[12] Lingxi Xie and Alan Yuille. Genetic cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1388–1397, 2017.

[13] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1980.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[15] Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S Bernstein, Alex Berg, and Li Fei-Fei. Scalable multi-label annotation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3099–3102, 2014.

[16] Rubin Daniel L Ertosun, Mehmet Günhan. Probabilistic visual search for masses within mammography images using deep learning. pages 1310–1315, 2015.

[17] Carneiro Gustavo Bradley Andrew P Dhungel, Neeraj. Automated mass detection in mammograms using cascaded deep learning and random forests. In *2015 international conference on digital image computing: techniques and applications (DICTA)*, pages 1–8. IEEE, 2015.

[18] Zhang Xiaoyong Homma Noriyasu Ichiji Kei Sugita Norihiro Kawasumi Yusuke Ishibashi Tadashi Yoshizawa Makoto Suzuki, Shintaro. Mass detection using deep convolutional neural network for mammographic computer-aided diagnosis. In *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 1382–1386. IEEE, 2016.

[19] Daniel Lévy and Arzav Jain. Breast mass classification from mammograms using deep convolutional neural networks. *arXiv preprint arXiv:1612.00542*, 2016.

[20] Aly Moustafa H. Salama, Wessam M. Deep learning in mammography images segmentation and classification: Automated CNN approach. *Alexandria Engineering Journal*, 60(5):4701–4709, October 2021. Publisher: Elsevier.

[21] Stuart Harvey Rubin. On randomization and discovery. *Information Sciences*, 177(1):170–191, 2007.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[23] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[24] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.