Neural Architecture Search (NAS) for Designing Optimal Power Quality Disturbance Classifiers

Qianchao Wang, Itamar Kapuza, Dmitry Baimel, Juri Belikov, Yoash Levron, Ram Machlev

***Abstract*—Deep learning techniques have recently demonstrated outstanding success when used for Power Quality Disturbance (PQD) classification. However, a core obstacle is that deep neural networks (DNN)s are complex models, and their architecture is designed using trial and error processes. Accordingly, the problem of finding the optimal architecture can be considered as a problem that consists of high-dimensional solutions. Meanwhile, in the last couple of years, Neural Architecture Search (NAS) techniques have been developed to efficiently find the best possible performance architecture for a specific task. In this light, the goal of this research is to develop a method to find optimal PQD classifiers using the NAS technique, based on an evolutionary algorithm. This method can converge efficiently to an optimal DNN architecture. Thus, a classifier that achieves high accuracy for PQDs classification is provided using limited resources and with minimal human intervention. This idea is demonstrated on two different DNN typologies- convolutional neural networks (CNN) and Bi-directional long short-term memory (Bi-LSTM). By adopting this method, the results of the generated PQD classifiers are more accurate when compared to recently developed classifiers.**

***Keywords*—Deep-Learning, Genetic algorithm, NAS, Neural architecture search, Power quality, PQD.**

I. Introduction

POWER quality is a measure of the degree to which voltage and current waveforms comply with established specifications [1]. Several power quality measures are harmonic distortions, variations in peak voltage, spikes and sags in voltages and currents, and variations in frequency [2]. In the last decade, Power Quality (PQ) monitoring tools are becoming a necessity [3]. One reason for the popularity of such tools is the continuing integration of nonlinear generators and loads in power grids, most of them based on power electronics technology, which may inject high-order voltage and current harmonics into the grid [4], [5].

In this light, many recent studies have focused on the detection and classification of Power Quality Disturbances (PQD)s. Up until the year 2018, most of the proposed algorithms merge feature extraction methods that are based on signal processing techniques with classification methods that stem from the theory of machine learning. One familiar approach is to use the Wavelet transform with a support-vector machine classifier [6]. Another well-known technique is to use the S-transform combined with different classifiers [7]. Other signal processing techniques that are being used are the fast Fourier transform [8] and sparse signal decomposition methods [9]. Moreover, in the last few years, with the evolution of Deep Learning (DL) techniques, better classifiers are now available in multiple fields and applications [10]. As part of this development, many works focusing on PQD classification such as [11]–[16] have implemented deep learning techniques, which in certain cases outperform the traditional classification algorithms. All of the above mentioned works seem to have advantages compared to traditional algorithms in terms of efficiency, noise immunity, and accuracy. Recently, a comprehensive review of deep learning for power quality has been published [17]. This survey aims to introduce deep learning to the power quality community by reviewing the latest applications and discussing the open challenges of using DL for PQD. Furthermore, review [17] also covers traditional algorithms of signal processing and feature extraction and compare them to DL approaches.

However, despite their evident success, deep neural networks (DNN) are very complex models since their architecture is designed using trial and error processes and they may consist of hundreds of layers and millions of parameters [18]. Accordingly, the problem of finding the optimal architecture can be considered as a problem that consists of high dimensional solutions. In addition, the development of DL models is computationally intensive with no guarantees on compute limits and performance, thus power experts may limit their practical usefulness [18].

Considering the above gap, the aim of this research is to develop an efficient method with low-complexity to find optimal PQD classifiers using Neural Architecture Search (NAS) technique [19]. NAS techniques can find a DNN model with high accuracy from large search space of possible network typologies and operations using limited resources and with minimal human intervention. The suggested method is based on evolutionary algorithm [20], [21] and operates as follows: first, a few random architectures for PQD classifier are generated. Then, at each iteration, these architecture are trained and ranked based on an accuracy metric using the testing data. At the final stage, the candidate architectures are updated based on a specific mechanism, which is inspired by

R. Machlev, I. Kapuza and Y. Levron are with the Andrew and Erna Viterbi Faculty of Electrical & Computer Engineering, Technion—Israel Institute of Technology, Haifa, 3200003, Israel (ramm@campus.technion.ac.il, itamarkapuza@campus.technion.ac.il, yoashl@ee.technion.ac.il).

Q. Wang is with the Key Laboratory of Energy Thermal Conversion and Control of Ministry of Education, School of Energy and Environment, Southeast University, Nanjing, 210096, China (230208053@seu.edu.cn).

D. Baimel is with the Shamoon College of Engineering, Beer-Sheva 84100, Israel (dmitrba@sce.ac.il).

J. Belikov is with the Department of Software Science, Tallinn University of Technology, Akadeemia tee 15a, 12618 Tallinn (juri.belikov@taltech.ee).

The work of Y. Levron was partly supported by Israel Science Foundation, grant No. 1227/18. The work of J. Belikov was partly supported by Estonian Research Council grant PRG1463.

the process of natural selection, allowing to search and obtain new architectures in the next iterations. This process continues until reaching a certain condition. It is shown that the method converges efficiently to an optimal architecture. Thus, a classifier which achieve high accuracy for PQDs classification is provided using limited resources and with minimal human intervention. This idea is demonstrated on two different neural network topologies - convolutional neural networks (CNN) and Bi-directional long short term memory (Bi-LSTM). The suggested NAS technique is tested on synthetic data-set and compared to reference classifiers from work [22].
Note that, as far as we know, the use of NAS for power quality classifiers and energy applications, is still new and no studies have been published so far.

II. NAS Method for PQD Classifiers

A. NAS - General

Neural Architecture Search (NAS) is an automatic architecture engineering process aiming to build a DL model with the best performance on a specific task while using limited resources and minimal human intervention. NAS was already proven to outperform manually designed DL architectures [23], [24], and has significant overlap with hyperparameter optimization [25]. In NAS the search space defines which architectures can be represented including different properties and hyperparameters of the model. Such parameters are number of layers, type of layers (e.g., convolution, fully connected, recurrent, pooling), activation functions (e.g., ReLU, softmax), type of optimizer (e.g., RMSProp, Adagrad, Adam), learning rate, and batch size. Furthermore, incorporating prior knowledge about typical properties of architectures well-suited for a task can reduce the search space size and simplify the search. In our case, properties of known DL-based PQD classifiers are used to define the search space. The search strategy describes how to explore the search space of the neural architectures. From one hand, it is desirable to find well-performing architectures quickly, while on the other hand premature convergence to a region of sub-optimal architectures should be avoided. Nowadays, different search strategies for NAS have been introduced, including random search, Bayesian optimization, evolutionary methods, reinforcement learning, and gradient-based techniques.

B. Evolutionary Search Strategy

Evolutionary algorithms are population-based metaheuristic optimizers. These algorithms consist of several essential components inspired by biological evolution [25]. An evolutionary algorithm starts with initialization, the first generation of population, followed by the subsequent steps until termination:

1) Select parents from the population for reproduction.
2) Apply recombination and mutation to create new classifiers.
3) Evaluate the fitness of the new classifiers.
4) Select the survivors of the population.

Recombination and mutation operators are chosen to balance population diversity and similarity. Then, survivor selection enables competition among the best classifiers of the population based on the chosen fitness function. First, parent architectures are selected for recombination and mutation, leading to a subset of network architectures in the search space. Then, each classifier, i.e. network architecture, is then trained and evaluated for fitness. Since recombination and mutation steps cause population growth, there is a need for the survivor selection component, which seeks to reduce the population size and enable competition among the classifiers. Several policies are used to achieve this, ranging from selecting only the best (elitist selection) to selecting all classifiers.
In this work, evolutionary algorithm named genetic algorithm (GA) is used for NAS [26]. In GA the classifiers are represented using a fixed-length encoding called the genome. A genome consists of genes, each gene contains a property or hyperparameter of the network.

C. Implementation of NAS for PQD

The suggested framework of NAS for PQD is described in Fig. 1. In addition, Table I presents the layers' properties of the corresponding genes, where the options $\{0, 1\}$ mean whether the mentioned property is used or not. In this way, each network is represented by a fixed-length genome encoding information about the network's architecture. Based on this table, an exmaple which illustrates a genome representation for CNN classifier is shown in Fig. 2. In this figure, each gene describes a specific layer's property (1D convolution or dense) and the final gene is the optimizer parameters.

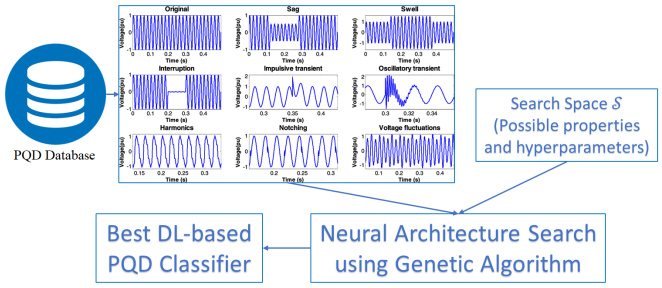


Fig. 1: Framework of proposed NAS for PQD classifier.

TABLE I: Properties of Layers.

1D-Conv (pair)	**Bi-LSTM** (pair)	**Dense**
Active$\in \{0, 1\}$	Active $\in \{0, 1\}$	Active $\in \{0, 1\}$
Filters$\in NCF$	Units $\in NBLU$	Nodes $\in NDN$
Batch normalization $\in \{0, 1\}$	Batch normalization $\in \{0, 1\}$	Batch normalization $\in \{0, 1\}$
Activation$\in AF$	–	Actication $\in AF$
Max pooling$\in \{0, 1\}$	–	–

The NAS for PQD process can be summarized in the form of Algorithm 1. Initialization of the first generation is done using a uniform probability on the search space $\mathcal{S}$. The fit function that is used to evaluate each network is cross-entropy. The selection operator used is rank selection, where in each generation the best $\beta\%$ of the classifiers are kept for the

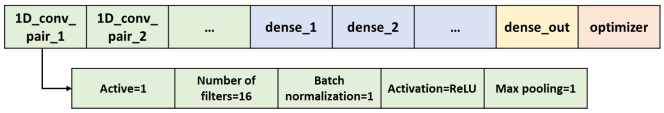


Fig. 2: GA genome which represents a CNN-based PQD classifier, including zoom in on the genes representing the properties of the first pair of 1D convolution layers

next generation, thus $(100-\beta)\%$ of the next generation are the result of reproduction from classifiers of the previous generation. Single-point crossover is used as the reproduction operator, and the mutation operator is done by changing one or several genes of an offspring using uniform sampling on the proper search space. The number of optional mutations increases during the evolution process.

Algorithm 1: NAS for PQD using Genetic Algorithm

Input:
$\mathcal{D}$: PQD dataset,
$\mathcal{S}$: search space,
G: number of generations,
N: number of classifiers in each generation.
Initialization: generating a set of randomized classifiers $\{M_{G=1,n}\}_{n=1}^{N}$ from $\mathcal{S}$, and training and tesing their PQD classification accuracy on $\mathcal{D}$ using cross-entropy.
for $g = 1, \ldots, G$ **do**
  Selection: producing a new generation $\{M'_{g,n}\}_{n=1}^{N}$ with a rank performance process on $\{M_{g-1,n}\}_{n=1}^{N}$: keeping the best $\beta\%$ architectures and reproduce $(100-\beta)\%$ architectures.
  Recombination: performing single-point crossover with uniform probability according to the genome length.
  Mutation: for each classifier in the new generation $\{M_{g,n}\}_{n=1}^{N}$, performing mutation on $\max\{0.3 \cdot N, \lfloor \frac{g}{4} \rfloor\}$ genes using uniform sampling on the proper options from $\mathcal{S}$.
  Evaluation: for each classifier in $\{M_{g,n}\}_{n=1}^{N}$, training and testing its PQD classification accuracy on $\mathcal{D}$ using cross-entropy loss function.
end
Output a set of classifiers in the final generation $\{M_{G,n}\}_{n=1}^{N}$ with their PQD classification accuracies.

Also, Table II presents the search space for both CNN and Bi-LSTM PQD classifiers.

TABLE II: Search Space: Architecture Parameters.

Parameter	Description
NCL	Number of 1D convolutional layers' pairs
NCF	Number of filters
NBLL	Number of Bi-LSTM layers' pairs
NBLU	Number of units in a Bi-LSTM
NDL	Number of dense layers
NDN	Number of nodes in a dense layer
AF	Activation functions
OPT	Number of optional optimizers

In this work the next definition for comparing computational complexity between brute force search and NAS is suggested:

$$\begin{aligned}\text{complexity ratio} &= \frac{\mathcal{O}(\text{solutions(brute force)})}{\mathcal{O}(\text{solutions(NAS)})} \\ &= \frac{OPT \cdot L \cdot (NDL \cdot NDN \cdot AF)}{N \cdot G},\end{aligned} \tag{1}$$

where $L = NBLL \cdot NBLU$ for Bi-LSTM classifier and $L = NCL \cdot NCF \cdot AF$ for CNN classifier.

III. Representative Examples Based on Different Neural Network Typologies

A. Dataset and the reference classifiers for PQD

The data-set is based on work [22], a PQDs Signals Generator. In this work, sixteen PQDs are generated and used by the Synthetic Generator shown in Table III. Our generated dataset contains 11200 signals, which are around 700 signals for each disturbance type. Each disturbance has a predefined number of copies with random parameters and additive white Gaussian noise.

TABLE III: PQDs Used by the Synthetic Generator.

Disturbances			
1	Normal	9	Notch (periodic)
2	Sag	10	Spike
3	Swell	11	Sag with harmonics
4	Interruption	12	Swell with harmonics
5	Harmonics	13	Interruption with harmonics
6	Flicker	14	Flicker with harmonics
7	Oscillatory transient	15	Flicker with sag
8	Impulsive transient	16	Flicker with swell

Four reference classifiers were used for comparison with the architecture generated by the NAS method. These reference classifiers are taken from [22], [27] and are based on different neural network typologies. The first network is convolutional neural network (CNN) shown in Table IV including 1-D convolution with a Rectified Linear Unit (ReLU), maxpooling layers, batch normalization layers and fully-connected layers which are designed to capture 1-D features. The kernel size is 3 and the stride value is 1 for both convolutional and maxpooling layers. The second network is Bi-directional long short-term memory (Bi-LSTM), shown in Table V which consists of Bi-LSTM and fully-connected layers to processes sequential inputs. Each Bi-LSTM layer contains 64 hidden units.The last two networks are CNNs named ResNet and Inception-Residual neural networks. The ResNet is based on [28], and uses Residual layers as shown in Fig. 3a. The architecture of this network is presented in Table VI. In this CNN, for all convolutional layers the kernel size is 16 and the stride value is 1 and for all max-pooling layers the kernel size is 2 and the stride value is 2. The Inception-Residual neural network is based on [29] which uses Inception-Residual blocks as shown in Fig. 3b. The architecture of this network is presented in Table VII. The implementation of the reference classifiers are done by 'Keras' and 'Tensorflow' frameworks in Python.

TABLE IV: Baseline CNN Architecture.

#	Layer	Parameters	Activation
1	Convolution (1x3)	Filter size = 32, Stride = 1	ReLU
2	Convolution (1x3)	Filter size = 32, Stride = 1	ReLU
3	MaxPooling (1x3) , Stride = 1		
4	Batch normalization		
5	Convolution (1x3)	Filter size = 64, Stride = 1	ReLU
6	Convolution (1x3)	Filter size = 64, Stride = 1	ReLU
7	MaxPooling (1x3) , Stride = 1		
8	Batch normalization		
9	Convolution (1x3)	Filter size = 128, Stride = 1	ReLU
10	Convolution (1x3)	Filter size = 128, Stride = 1	ReLU
11	Global MaxPooling		
12	Batch normalization		
13	Flatten		
14	Fully connected	Size = 128	ReLU
15	Fully connected	Size = 128	ReLU
16	Batch normalization		
17	Fully connected	Size = 16	Softmax

TABLE V: Baseline Bi-LSTM Architecture.

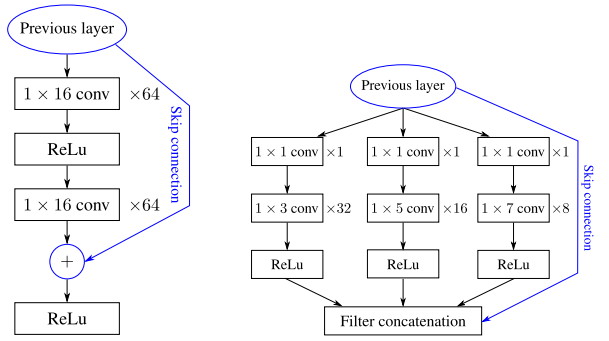
#	Layer	Parameters	Activation
1	Bi-LSTM	hidden units = 64	
2	Bi-LSTM	hidden units = 64	
3	Bi-LSTM	hidden units = 64	
4	Fully connected	Size = 128	ReLU
5	Fully connected	Size = 128	ReLU
6	Fully connected	Size = 128	ReLU
7	Fully connected	Size = 16	Softmax

TABLE VI: ResNet Architecture [28]

#	Layer	Parameters	Activation
1	Convolution (1x16)	Filter size = 64, Stride = 1	ReLU
2	MaxPooling (1x2)	Stride = 2	–
3		Residual block	
4		Residual block	
5	MaxPooling (1x2)	Stride = 2	–
6		Residual block	
7		Residual block	
8	MaxPooling (1x2)	Stride = 2	–
9	Fully connected	Size = 256	ReLU
10	Fully connected	Size = 16	Softmax

TABLE VII: Inception-Residual Architecture [29]

#	Layer	Parameters	Activation
1		Inception-Residual block	
2		Inception-Residual block	
3		Inception-Residual block	
4		Inception-Residual block	
5		Inception-Residual block	
6	fully connected	size = 256	dropout = 0.1
7	fully connected	size = 16	Softmax



(a) Residual block used for CNN-3

(b) Inception-Residual block used for CNN-4

Fig. 3: Special Convolution layers that were used in this work based on [28] and [29].

For all networks the fitting parameters are shown in Table VIII. The size of each mini-batch is set to 64 signals, which are selected randomly for the back-propagation stage in the training process. The number of epochs, which indicates the number of times the network run on the entire training dataset, is 50 for the CNN, ResNet and Inception-Residual, and 60 for the Bi-LSTM. The selected loss function is based on cross-entropy, and the optimizer is "Nadam" for the CNN, and "Adam" for the Bi-LSTM. The built-in 'ReduceLROnPlateau' is utilized in both classfiers to reduce learning rate.

TABLE VIII: Fitting Parameters for reference classifiers.

Parameters	Values
Batch size	64
Epcoch	50 (CNNs), 60 (Bi-LSTM)
Loss function	Cross-entropy
Optimizer	'Nadam'

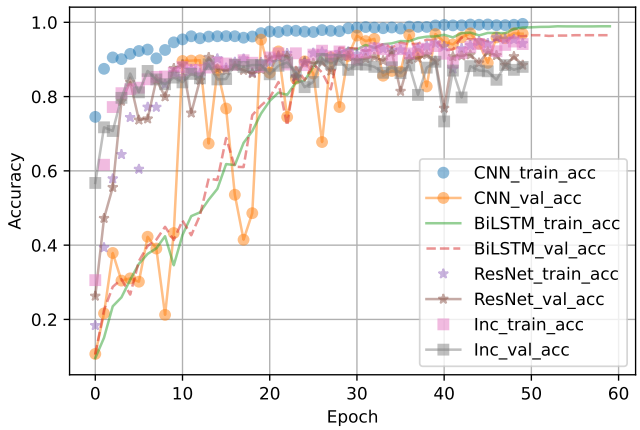
The training set includes 90% of the samples in the complete data-set, and the testing set includes the remaining 10%. The accuracy is defined as:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}. \quad (2)$$

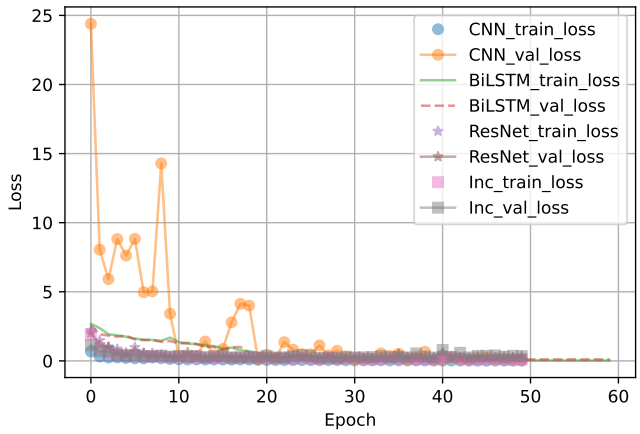
The training process for the reference classifiers is shown in Fig. 4 and the confusion matrices representing the classification results are shown in Fig. 5. The accuracy of CNN, Bi-LSTM, ResNet and Inception-Residual is 93.5%, 90.41%, 88.48%, 89.64% respectively. Although the accuracy of both classifiers tends to be 1 in training process, there is some bias when the classifiers are validated. From the confusion matrix it can be seen that the trained networks have errors when classify the flicker as spike . There are also other errors in both classifiers which means they can be further improved.

B. Results of NAS for PQD

Although the the reference classifiers have good results, the structure of networks might not be optimal in terms of accuracy. The performance of the classifiers can be improved if better network architecture can be found using the suggested



(a) Accuracy of the reference classifiers



(b) Loss of the reference classifiers

Fig. 4: Accuracy and Loss of the reference classifiers.

NAS method from Section. II as presented next. Basline CNN and Bi-LSTM are chosen to optimize the neural architecture using NAS. Each convolutional module has two 1-D convolutional layers with activation functions: a maxPooling layer and a batchnormalization layer. Each Bi-LSTM module has a Bi-LSTM layer and a batchnormalization layer. Each fully-connected module has a dense layer with an activation function and a batchnormalization layer. For CNN, the optimized parameters contain the number of filters in convolutional layers, the number of hidden units in dense layers, the types of activation, and the batchnormalization layers. For Bi-LSTM, the optimized parameters contain the number of hidden units in both Bi-LSTM layers and dense layers, the types of activation functions, and the batchnormalization layers. The boundaries of the optimized parameters are shown in Table IX. The number of filters and hidden units is selected according to the power of 2. During the optimization process, the modules of layers will first be confirmed whether they are active or not. The numbers of generations and off-springs are $[G = 10, N = 10]$ for CNN and $[G = 5, N = 5]$ for Bi-LSTM and for both network types $\beta = 20\%$. The other fit parameters are the same as the reference classifiers, expect the optimizer which will be optimized by NAS. The convergence of NAS optimization is shown in Fig. 6 and the optimal structures searched by NAS are shown in Table X and Table XI for CNN and Bi-LSTM. Using these parameters it can be seen that the complexity ratio (1) is bigger than 4500 for CNN and bigger than 2500 for LSTM.

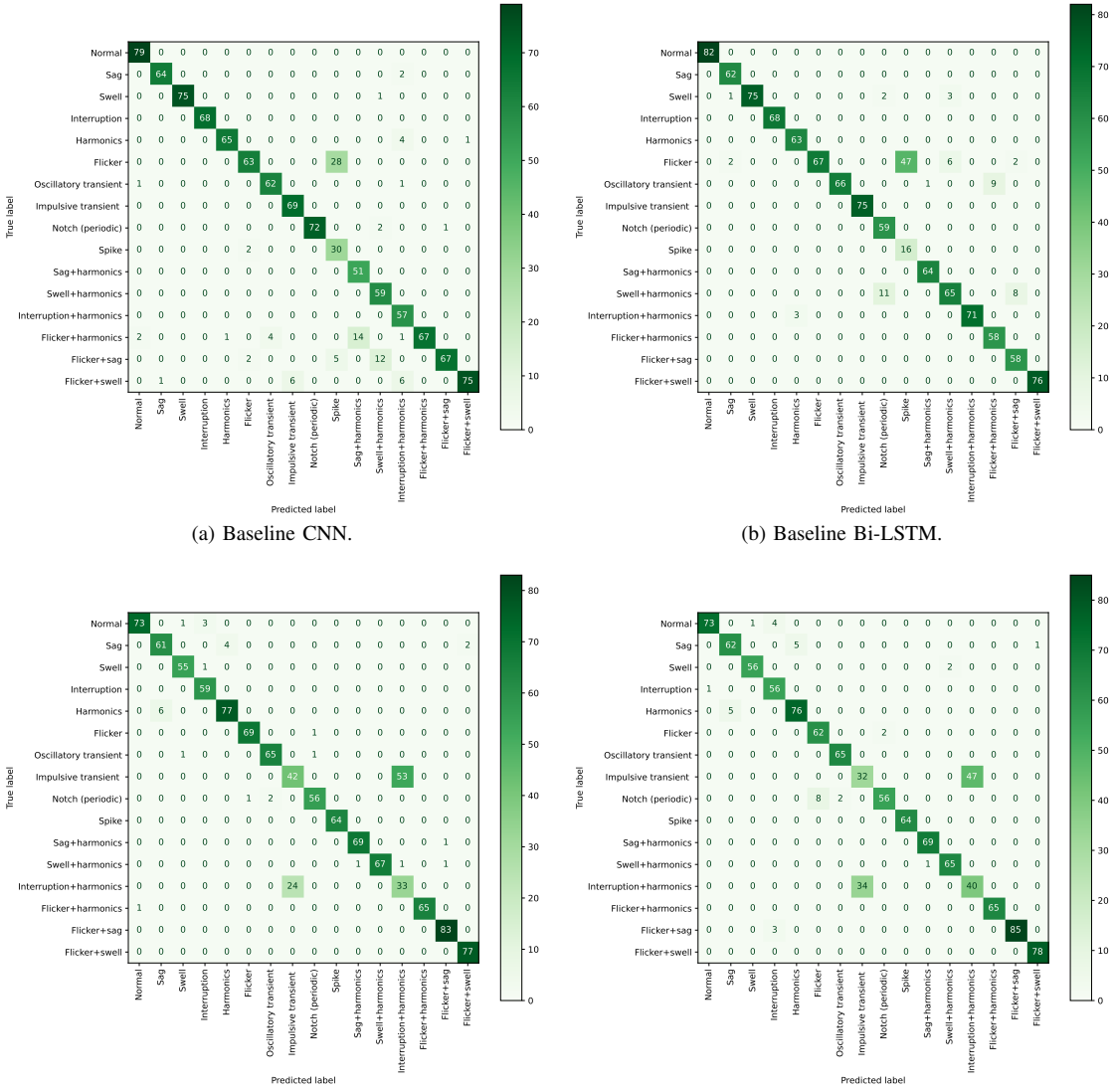
TABLE IX: The Boundaries of Optimized Parameters.

Parameters	**Boundaries**
NCF	$[2^0; 2^7]$
NBLU	$[2^0; 2^8]$
NDN	$[2^0; 2^8]$
AF	[‘linear’,‘Relu’,‘sigmoid’,‘tanh’]
Batch normalization	$0; 1$
NCL	10
NBLL	5
NDL	10
OPT	[‘adam’,‘rmsprop’,‘adagrad’,‘adadelta’]

TABLE X: Optimal CNN Architecture using NAS.

Module	Layer	Parameters	Activation
1	Convolution (1x3)	Filter size = 16, Stride = 1	Sigmoid
	Convolution (1x3)	Filter size = 16, Stride = 1	Sigmoid
	MaxPooling (1x2) , Stride = 1		
	Batch normalization		
2	Convolution (1x3)	Filter size = 128, Stride = 1	Sigmoid
	Convolution (1x3)	Filter size = 128, Stride = 1	Sigmoid
	Batch normalization		
3	Convolution (1x3)	Filter size = 8, Stride = 1	Relu
	Convolution (1x3)	Filter size = 8, Stride = 1	Relu
	MaxPooling (1x2) , Stride = 1		
4	Convolution (1x3)	Filter size = 16, Stride = 1	Tanh
	Convolution (1x3)	Filter size = 16, Stride = 1	Tanh
	Batch normalization		
5	Convolution (1x3)	Filter size = 8, Stride = 1	Relu
	Convolution (1x3)	Filter size = 8, Stride = 1	Relu
	MaxPooling (1x2) , Stride = 1		
6	Convolution (1x3)	Filter size = 64, Stride = 1	Relu
	Convolution (1x3)	Filter size = 64, Stride = 1	Relu
	MaxPooling (1x2) , Stride = 1		
	Batch normalization		
7	Flatten		
8	Fully connected	Size = 32	Tanh
9	Fully connected	Size = 16	Tanh
10	Fully connected	Size = 64	Sigmoid
11	Fully connected	Size = 64	Tanh
	Batch normalization		
12	Fully connected	Size = 16	Softmax

The average classification accuracy of the testing sets for the CNN and Bi-LSTM classifiers using NAS are 98.59% and 96.79%, respectively as can be seen from Table XII. For CNN using NAS, note that the ‘Relu’ is not the only activation function since ‘Sigmoid’ and ‘Tanh’ are used as well (although they considered to be less common for CNNs). For Bi-LSTM using NAS, the optimized structure are similar to the baseline except considering the usage of batchnormalization layers in the structure. Both the optimized models use ’Adam’ as the optimizer. The confusion matrices of both optimized CNN



(a) Baseline CNN.

(b) Baseline Bi-LSTM.

(c) ResNet.

(d) Inception-Residual.

Fig. 5: Confusion Matrix of reference classifiers.

TABLE XI: Optimal Bi-LSTM structure using NAS.

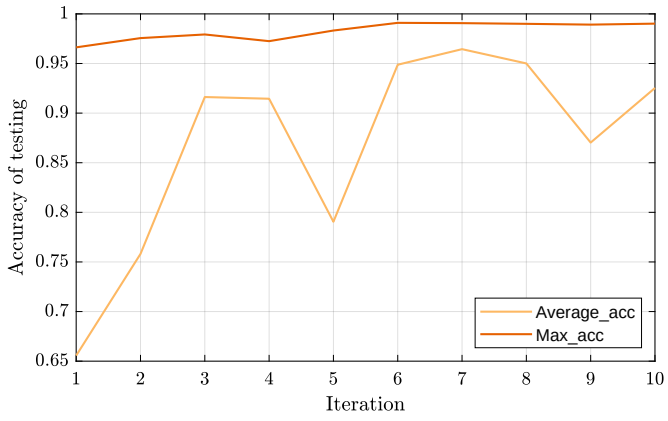
Module	Layer	Parameters	Activation
1	Bi-LSTM	hidden units = 128	
2	Bi-LSTM	hidden units = 128	
	Batch normalization		
3	Bi-LSTM	hidden units = 256	
4	Bi-LSTM	hidden units = 256	
4	Fully connected	Size = 256	Relu
5	Fully connected	Size =128	Relu
6	Fully connected	Size =256	Relu
7	Fully connected	Size = 16	Softmax

TABLE XII: Simulation Average Accuracy Results.

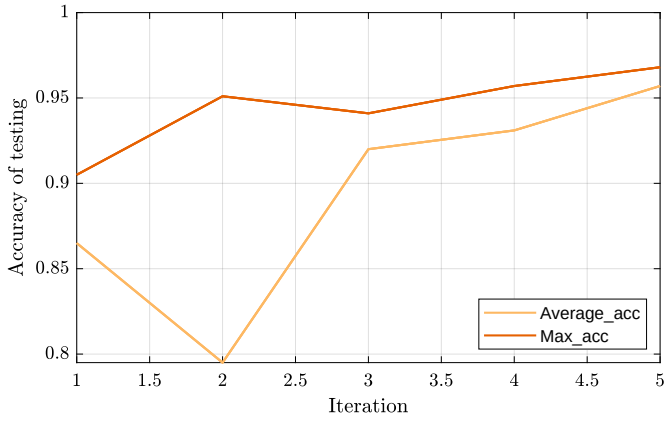
	Training	Testing
CNN baseline	97.97%	93.5%
CNN with NAS	**99.09%**	**98.59%**
Bi-LSTM baseline	96.41%	90.41%
Bi-LSTM with NAS	**98.83%**	**96.79%**
ResNet	95.35%	88.48%
Inception-Residual	94.17%	89.64%

and Bi-LSTM are shown in Fig. 7. It can be seen that when comparing the classifiers optimized by NAS with the reference classifiers, there are less errors.

In addition, the computational complexity of CNN and BiLSTM baseline models and the classifiers optimized by brute force procedures are reported in Table XIII, based on the search space defined in Table IX according to the parameters in Table II. Also, the computational complexity of NAS procedure is included, based on number of generations and population in each generation. It can be seen that the NAS procedure for finding the optimal classifier saved a significant amount of training time compared to brute force, which can be also described by complexity ratio defined in eq (1).



(a) NAS CNN

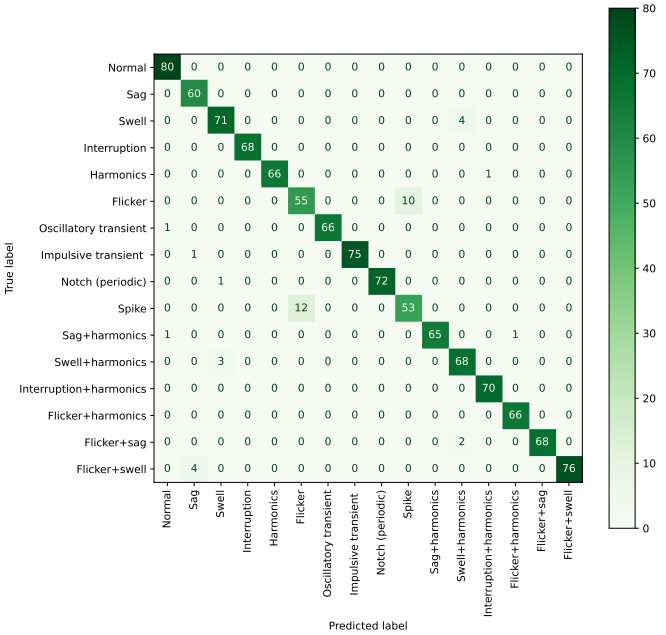


(b) NAS Bi-LSTM

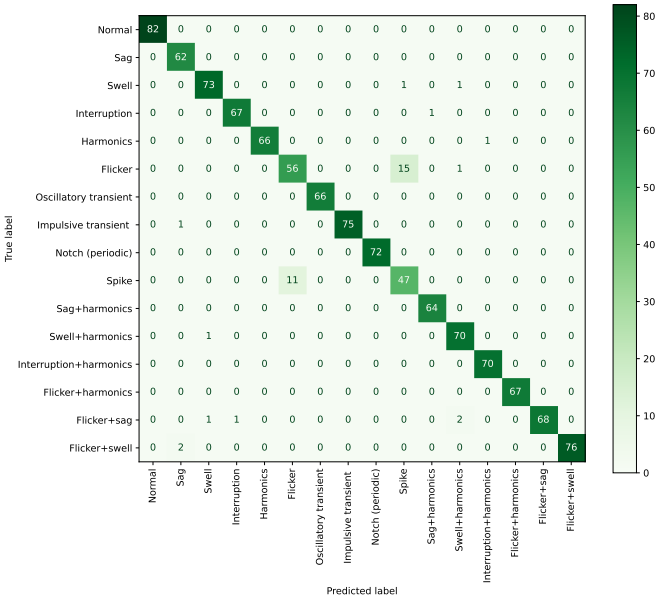
Fig. 6: The convergence of NAS optimization through iterations (generations).

TABLE XIII: Training Time Results.

Architecture	**Scenario**	Computational complexity
CNN	Baseline	1
	Brute force	460800
	NAS	100
BiLSTM	Base-line	1
	Brute force	64800
	NAS	100



(a) NAS CNN



(b) NAS Bi-LSTM

Fig. 7: The confusion matrices by NAS optimization.

IV. Conclusion

In the last decade, many studies focus on classification of power quality disturbances using deep neural networks (DNNs). However, despite their high performance, DNNs are complex models and their architecture, which consist of hundreds of layers and millions of parameters, is designed using trial and error processes. Furthermore, training such models is considered computationally intensive with no guarantees on compute limits and performance. In this light, this research suggests a method to find optimal PQD classifiers efficiently using Neural Architecture Search (NAS) technique, based on genetic algorithm. By using NAS an architecture of high-accuracy PQD classifier is generated efficiently using limited resources and with minimal intervention. This idea is demonstrated on two different neural network typologies-CNN and Bi-LSTM. From the results, it can be seen that the accuracy of the proposed method has increased by more than 5% when compared to state-of-the-art classifiers. Another key result is that the suggested NAS method operating with lower computational complexity and convergences faster to the optimal solution when compared to brute force search.

More broadly, we believe that NAS techniques have high potential to improve the performance of deep learning algorithms, which are increasingly being used nowadays in the power system community. Therefore, there may be significant room for future research, which may focus on the use of NAS techniques in other energy related applications.

REFERENCES

[1] M. H. J. Bollen, Understanding Power Quality Problems. John Wiley & Sons, 1999.

[2] C. Sankaran, Power Quality. CRC Press, 2017.

[3] M. H. J. Bollen and I. Y.-H. Gu, Signal Processing of Power Quality Disturbances. John Wiley & Sons, Inc., 2006.

[4] E. Hossain, M. R. Tur, S. Padmanaban, S. Ay, and I. Khan, "Analysis and mitigation of power quality issues in distributed generation systems using custom power devices," IEEE Access, vol. 6, pp. 16 816–16 833, 2018.

[5] S. K. Khadem, M. Basu, and M. Conlon, "Power quality in grid connected renewable energy systems: role of custom power devices," Renewable Energy and Power Quality Journal, vol. 1, no. 08, pp. 878–881, 2010.

[6] D. D. Yong, S. Bhowmik, and F. Magnago, "An effective power quality classifier using wavelet transform and support vector machines," Expert Systems with Applications, vol. 42, no. 15-16, pp. 6075–6081, 2015.

[7] R. Kumar, B. Singh, D. T. Shahani, A. Chandra, and K. Al-Haddad, "Recognition of power-quality disturbances using s-transform-based ANN classifier and rule-based decision tree," IEEE Transactions on Industry Applications, vol. 51, no. 2, pp. 1249–1258, 2015.

[8] F. A. S. Borges, R. A. S. Fernandes, I. N. Silva, and C. B. S. Silva, "Feature extraction and power quality disturbances classification using smart meters signals," IEEE Transactions on Industrial Informatics, vol. 12, no. 2, pp. 824–833, 2016.

[9] M. S. Manikandan, S. R. Samantaray, and I. Kamwa, "Detection and classification of power quality disturbances using sparse signal decomposition on hybrid dictionaries," IEEE Transactions on Instrumentation and Measurement, vol. 64, no. 1, pp. 27–38, 2015.

[10] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[11] N. Mohan, K. P. Soman, and R. Vinayakumar, "Deep power: Deep learning architectures for power quality disturbances classification," in 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), 2017.

[12] H. Liu, F. Hussain, Y. Shen, S. Arif, A. Nazir, and M. Abubakar, "Complex power quality disturbances classification via curvelet transform and deep learning," Electric Power Systems Research, vol. 163, pp. 1–9, 2018.

[13] Y. Deng, L. Wang, H. Jia, X. Tong, and F. Li, "A sequence-to-sequence deep learning architecture based on bidirectional GRU for type recognition and time location of combined power quality disturbance," IEEE Transactions on Industrial Informatics, vol. 15, no. 8, pp. 4481–4493, 2019.

[14] S. Wang and H. Chen, "A novel deep learning method for the classification of power quality disturbances using deep convolutional neural network," Applied Energy, vol. 235, pp. 1126–1140, 2019.

[15] Y. Zhang, Y. Zhang, and X. Zhou, "Classification of power quality disturbances using visual attention mechanism and feed-forward neural network," Measurement, vol. 188, p. 110390, 2022.

[16] R. S. Salles and P. F. Ribeiro, "The use of deep learning and 2-d wavelet scalograms for power quality disturbances classification," Electric Power Systems Research, vol. 214, p. 108834, 2023.

[17] R. A. de Oliveira and M. H. Bollen, "Deep learning for power quality," Electric Power Systems Research, vol. 214, p. 108887, 2023.

[18] S. Chatzivasileiadis, A. Venzke, J. Stiasny, and G. Misyris, "Machine learning in power systems: Is it time to trust it?" IEEE Power and Energy Magazine, vol. 20, no. 3, pp. 32–41, 2022.

[19] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," The Journal of Machine Learning Research, vol. 20, no. 1, pp. 1997–2017, 2019.

[20] L. Xie and A. Yuille, "Genetic CNN," in Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 1379–1388.

[21] M. Suganuma, S. Shirakawa, and T. Nagao, "A genetic programming approach to designing convolutional neural network architectures," in Proceedings of the Genetic and Evolutionary Computation Conference, 2017, pp. 497–504.

[22] R. Machlev, A. Chachkes, J. Belikov, Y. Beck, and Y. Levron, "Open source dataset generator for power quality disturbances with deep-learning reference classifiers," Electric Power Systems Research, vol. 195, p. 107152, 2021.

[23] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," arXiv preprint arXiv:1611.01578, 2016.

[24] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Aging evolution for image classifier architecture search," in AAAI Conference on Artificial Intelligence, vol. 2, 2019, p. 2.

[25] M. Feurer and F. Hutter, "Hyperparameter optimization," in Automated Machine Learning. Springer, Cham, 2019, pp. 3–33.

[26] S. Mirjalili, "Genetic algorithm," in Evolutionary Algorithms and Neural Networks. Springer, 2019, pp. 43–55.

[27] R. Machlev, M. Perl, J. Belikov, K. Y. Levy, and Y. Levron, "Measuring explainability and trustworthiness of power quality disturbances classifiers using xai—explainable artificial intelligence," IEEE Transactions on Industrial Informatics, vol. 18, no. 8, pp. 5127–5137, 2021.

[28] H. Li, B. Yi, Q. Li, J. Ming, and Z. Zhao, "Evaluation of DC power quality based on empirical mode decomposition and one-dimensional convolutional neural network," IEEE Access, vol. 8, pp. 34 339–34 349, 2020.

[29] R. Gong and T. Ruan, "A new convolutional network structure for power quality disturbance identification and classification in micro-grids," IEEE Access, vol. 8, pp. 88 801–88 814, 2020.