Efficient and Mobile Deep Learning Architectures for Fast Identification of Bacterial Strains in Resource-Constrained Devices

Rafael Gallardo-García[1], Sofía Jarquín-Rodríguez[2], Beatriz Beltrán-Martínez[1], Rodolfo Martínez-Torres[1], and Carlos Hernández-Gracidas[3]

[1]Language and Knowledge Engineering Laboratory, Benemérita Universidad Autónoma de Puebla, México
[2]Faculty of Chemical Sciences, Benemérita Universidad Autónoma de Puebla, México
[3]Faculty of Physical and Mathematical Sciences, Benemérita Universidad Autónoma de Puebla, México

{rafael.gallardo, ana.jarquin}@alumno.buap.mx
{bbeltran,beetho}@cs.buap.mx
cahernandezgr@conacyt.mx

Abstract.

Keywords.

1 Introduction

Bacteria are small, unicellular microorganisms which are found almost everywhere on Earth. Most bacteria are not harmful to humans, because less than 1% of the different species of bacteria make people sick [18].

Due to the impact of bacteria in human life, the recognition of bacterial genera or species is a very common and important task in many areas such as medicine, veterinary science, biochemistry, food industry, or farming [38]. Traditional laboratory methods for the identification of bacterial strains commonly require an expert with knowledge and experience in the field. On the other hand, most techniques designed for rapid and automated identification of microbiological samples are based of biochemical or modular biology technologies [11, 33]. These traditional approaches are well established, however they are expensive and time consuming since they require complex sample preparation [2].

Automatizing the process of bacteria identification is very promising in the field of bioimage informatics. This field has yielded powerful solutions for specific image analysis tasks such as object detection, motion analysis or morphometric features [32]. Even so, most image analysis algorithms have been developed for very specific tasks or biological assays. Faster, more precise and less expensive computational approaches to bacterial strain classification are necessary.

This paper presents a comparison among several deep learning architectures when solving the task of bacterial strain classification. These architectures are similar in that they all have been proposed as efficient or mobile solutions for image classification, with the main advantage of low computational cost in terms of memory and number of calculations. In the experiments, the architectures were initialized with a pre-trained model (also called transfer-learning) over the ImageNet dataset [4], and then fine-tuned to our specific problem. The networks were trained from scratch in the cases where pre-trained models were unavailable. The evaluation of each proposal was performed using 10-fold cross validation over two different datasets: the original one, and a new one, generated by using data augmentation techniques. We present comparative Tables of all architectures, trained and evaluated over both datasets, reporting cross-validation averages of top-1 accuracy, top-5 accuracy, weighted precision, weighted recall, and weighted F1 score.

The main contributions of this work are summarized as follows:

— We demonstrate that our data augmentation techniques considerably improve the overall performance of several architectures, as compared to the use of the original dataset.

— We present proposals that reach and exceed several methods available in the literature when measuring top-1 classification accuracy.

— We carry out a complete and robust evaluation of each method, training and testing with 10-fold cross validation in both datasets, and measuring performance with five different metrics.

— This work indicates that near-state-of-the-art performance can be achieved without using extremely expensive networks: all of our proposals have less than 10 million parameters.

— The efficient and low-resource nature of the networks we used, makes it easily to adapt and scale them to new problems and datasets, also making implementation and deployment on mobile or embedded devices easy.

2 State of the art

Holmberg et al. [8] published the first attempt to classify bacterial strains in an automated way, they used classification trees to extract the most important features and then classified them with artificial neural networks, achieving an accuracy of 76% in a dataset with 5 species of bacteria.

Later, in 2001, Liu et al. [15] presented a computer-aided system to extract size and shape measurements of digital images of microorganisms to classify them into their appropriate morphotype. Their work aimed to classify automatically each cell into one of 11 predominant bacterial morphotypes. This classifier had an accuracy of 96% on a set of 1,471 cells and 97% on a set of 4,270 cells.

Trattner et al. [36] proposed an automatic tool to identify microbiological data types with computer vision and statistical modeling techniques. Their methodology provided an objective and robust analysis of visual data, to automatize bacteriophage typing methods.

Some works consist in the identification of just one species of bacteria. Forero et al. [5] presented a method for automatic identification of *Mycobacterium tuberculosis* by using Gaussian mixture models. The authors used geometrical and color features of the images.

In 2013, Ahmed et al. [2] published a light scatter-based approach to bacterial classification using distributed computing (due to the computational complexity of their feature extractors). They used Zernike and Chebyshev moments and Haralick texture features, then, the best features were selected using Fisher's discriminant. The classifier was a support vector machine with a linear kernel. The classification accuracy varied from 90% to 99% depending on the species.

The Digital Image of Bacterial Species (DIBaS) dataset was presented in 2017 [38], Zieliński et al. published a deep learning approach to bacterial colony classification [38]. In this work, the authors used Convolutional Neural Networks as feature extractors and support vector machines or random forests as classifiers. Their overall highest accuracy over the DIBaS dataset was 97.24%, achieved by Fisher vector [25] ALGO ANDA MAL AQUÍ, PUES AL INICIO DEL PÁRRAFO SE MENCIONA QUE EL DATASET ES DE 2017, PERO ESTA REFERENCIA SE SUPONE QUE ES DE 2007 (10 AÑOS ANTES DE QUE SE CREARA EL DATASET CON EL QUE SE PROBÓ) encoders combined with VGG-M [30] ESTE TAMBIÉN ES DE FECHA ANTERIOR. SUPONGO QUE EN REALIDAD EL DATASET ES DE 2007, PERO HABRÍA QUE VERIFICAR network.

In 2018, Faruk Nasip and Zengin Kenan [20] published another deep learning approach to solve the DIBaS classification task. The authors used two different architectures, which mainly consist in bacteria classification using a VGG Net-based [30] approach and an AlexNet-based [14] approach, achieving 98.25% and 97.53% of classification accuracy, respectively.

More recently, in 2019, three works over the DIBaS dataset were published. The first one, by Khalifa et al. [12], reached a 98.22% of classification accuracy with a deep convolutional neural network (CNN) and data augmentation techniques, which they called "Deep bacteria". The second one, by Muhammed Talo [34], outperformed "Deep bacteria", achieving a 99.2% of classification accuracy by fine-tuning a pre-trained model of ResNet-50. The third work from 2019 [26] reached a 95.09% of classification accuracy by fine-tuning a pre-trained version of the

MobileNet V2 architecture [27], and using some data augmentation techniques.

To the best of our knowledge, just three papers about bacteria classification over the DIBaS dataset were published in 2020. Elaziz et al. [1] achieved a 98.68% of classification accuracy, by using fractional-order orthogonal moments to extract features; also, they proposed a new feature selection method, which they called SSATLBO (Salp Swarm Algorithm + teaching-based learning optimization). In [7], authors achieved a 94.22% of classification accuracy, by using a fine-tuned version of a pre-trained MobileNet V2 architecture and data augmentation techniques. The last method from 2020, proposed by Satoto et al. [28], reached a 98.59% of classification accuracy by using a custom CNN topology and data augmentation techniques.

3 Traditional methods for bacterial identification

In clinical microbiology laboratories, manual and automated methods are used for bacterial identification. Classification of microorganisms to the species level is one of the most important tasks to microbiologists and other scientists involved in many areas. Phenotypic methods are the standard method to identify them. However, they have some limitations for certain types of microorganisms. Molecular methods and proteomics-based methods are complementary to phenotypic methods and allow to overcome some limitations, although their implementation is less usual in some laboratories due to the higher cost and experience required for its application.

3.1 Phenotypic methods

Traditional bacterial phenotypic identification include some macroscopic features and biochemical properties such as morphology, hemolysis, culture medium, production of certain metabolites, type of growth in relation to atmosphere, temperature and nutrition. Preliminary identification of most cultivable bacteria is based on such morphological characteristics [6]. Manual testing to identify bacteria requires from hours to several days and a large amount of biological material, so that most laboratories use either commercially available miniaturized biochemical test systems and the reading of these panels is usually done by automated systems and then incorporating the obtained data in a computer [37].

The following list contains some of the most common techniques that belongs to this category:

— Culture medium.

— Multi-test galleries.

— Automated systems.

3.2 Microscopy techniques

Optical microscopy is a simple and fast method that often guides to clinical diagnosis. However, this method is not successful by itself, usually because the identification of microorganisms is limited to by factors such as small cells, variation in size or resolution of the optical microscope. Visualization under the microscope is more specific and easier to perform with the use of fluorescent markers. [19].

The following list contains some of the most common techniques that belongs to this category:

— Transmission electron microscopy (TEM).

— Scanning electron microscopy (SEM).

— Confocal microscopy (CLSM).

3.3 Proteomics-based methods

Proteomics is the study and characterization of the set of proteins expressed by a genome (proteome) [3]. Methods based on mass spectrometry (MS) are important tools for microbial typing, they detect the mass-charge ratio (m/z) of the analyte. Particularly, Matrix Assisted Laser Desorption/Ionization-Time of Flight (MALDI-TOF) is a method that generates a peptide mass fingerprint (PMF) in the form of a spectral profile that is characteristic of each species, the result is compared with existing databases for its identification [31, 29].

3.4 Molecular methods

Molecular methods are complementary or alternative procedures, which provide high-throughput, more sensitive and discriminatory analyzes of microorganisms or genetic polymorphisms through the amplification and detection of specific nucleic acid targets. Some strains do not have a specific feature, the same strain can generate different results in repeated tests, most of the time molecular methods can solve these problems [37, 6].

The following list contains some of the most common techniques that belong to this category:

— Real-time PCR.

— Pulse-Field Gel Electrophoresis (PFGE).

— 16S-RNA.

3.5 Advantages and disadvantages of traditional methods

Table **??** summarizes the advantages and disadvantages of the traditional bacterial detection methods. The first column (from left to right) indicates the category of the technique, the second column shows some examples of commercial products that belong to each category, the third and fourth columns presents some advantages and disadvantages of each category, respectively.

4 Materials and methods

This section describes the methodologies we used to generate the dataset (the class distribution of the samples is also presented). Details about implementation, fine-tuning, training, and evaluation of all the deep learning architectures are provided as well.

4.1 Digital Images of Bacteria Species dataset

The original version of DIBaS dataset contains a total of 33 species of microorganisms, approximately 20 RGB images (2048×1532 pixels) per each species. We removed the set of images of *Candida albicans* colonies as it is considered fungi [21]. The dataset was collected by the Chair of Microbiology of the Jagiellonian University in Krakow. The samples were stained using the Gramm's method. All images were taken with an Olympus CX31 Upright Biological Microscope and an SC30 camera with a 100 times objective under oil-immersion [38]. The DIBaS dataset is publicly available[1].

4.2 Data augmentation

To make our experiments easily repeatable, we provide Algorithm 1, which shows the process we followed to augment the original DIBaS dataset:

Algorithm 1: Data augmentation process.

```
Result: An augmented version of the DIBaS
        dataset.
for Species in DIBaS do
    for Sample in Species do
        for Shape in
         [100,200,300,400,500,600,700] do
            L=Obtain 5 crops of Sample with
             size Shape;
            resize all images in L to 224x224
             px;
        end
        add all images in L to augmented
         folder;
        resize Sample to 224x224px;
        add Sample to augmented folder;
    end
end
```

Our data augmentation strategy is aimed to provide at least 35 different images per each sample in the original dataset. The proposed approach tries to simulate different levels of zoom for the same bacterial sample, which is achieved

[1] http://misztal.edu.pl/software/databases/dibas/

Table 1. Samples per species, both in original and augmented versions of the dataset.

Genera	Species	# Original	# Augmented
Acinetobacter	baumanii	20	709
Actinomyces	israelli	23	828
Bacteroides	fragilis	23	828
Bifidobacterium	spp.	23	828
Clostridium	perfringens	23	828
Enterococcus	faecalis	20	720
	faecium	20	720
Escherichia	coli	20	720
Fusobacterium	spp.	23	828
Lactobacillus	casei	20	720
	crispatus	23	720
	delbrueckii	23	720
	gasseri	23	720
	jehnsenii	23	720
	johnsonii	23	720
	paracasei	23	720
	plantarum	23	720
	reuteri	23	720
	rhamnosus	23	720
	salivarius	23	720
Listeria	monocytogenes	22	792
Micrococcus	spp.	21	756
Neisseria	gonorrhoeae	23	828
Porfyromonas	gingivalis	23	828
Propionibacterium	acnes	23	828
Proteus	spp.	23	720
Pseudomonas	aeruginosa	23	720
Staphylococcus	aureus	23	720
	epidermidis	23	720
	saprophiticus	23	720
Streptococcus	agalactiae	23	720
Veionella	spp.	22	792

by cropping different regions of different sizes of the full image. Specifically, we obtained five crops per image, with seven different sizes: from 100 to 700, with steps of 100, as we can see in Algorithm 1 (to see a graphical example, see Figure 2). All of the proposed architectures are designed to process inputs of 224x224 pixels; hence, after data augmentation, we resized every sample to the input size using Lanczos interpolation. In the end, our augmented version of DIBaS had a total of 24,073 samples (including the original images). Detailed information about sample distribution across classes is available in Table 1. Image 1 shows some sample images, generated with our data augmentation method.

4.3 Deep Learning architectures

This section contains a short description of each architecture we present in this paper, as well as details about their implementation, transfer learning and fine-tuning.

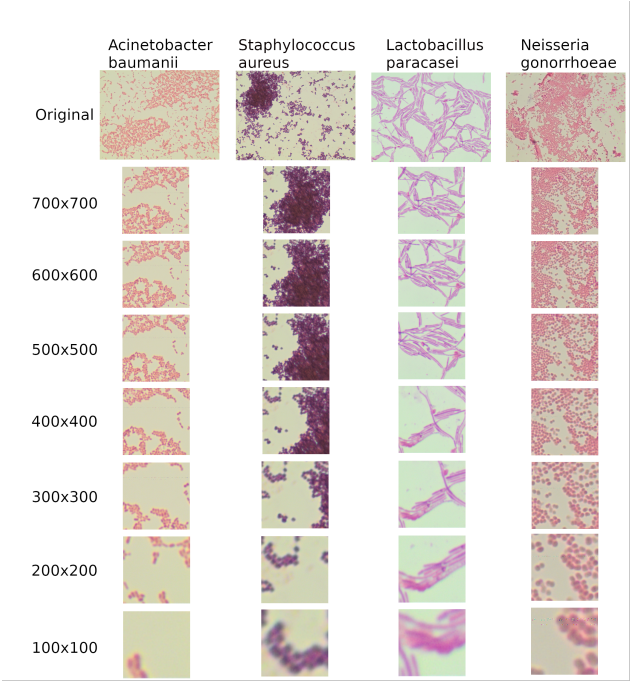

Fig. 1. Augmented samples of 4 of the 32 classes in the dataset. This figure shows the original image and one sample of each amplification.

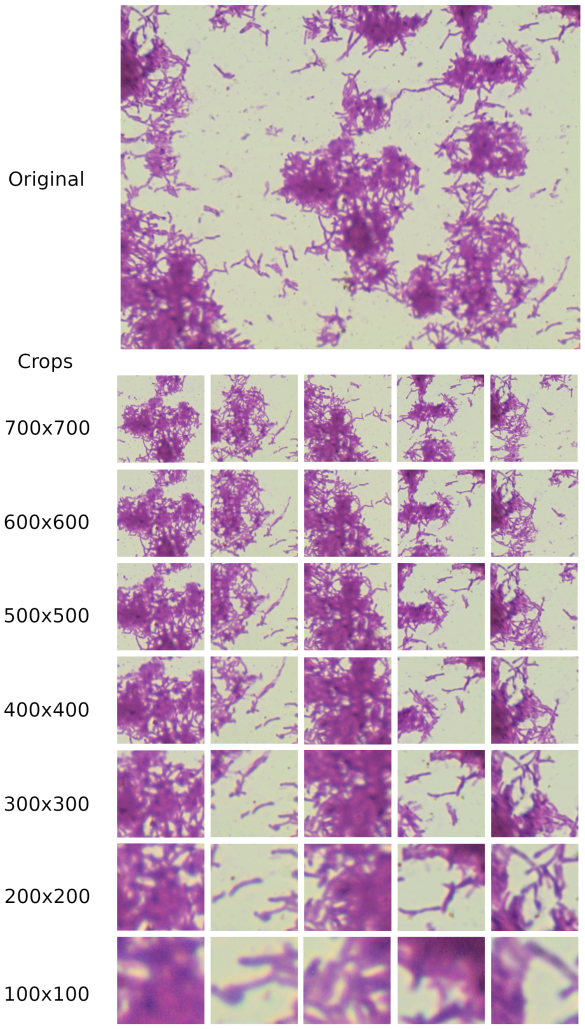

Fig. 2. Illustration of each sample generated by our data augmentation method (see Algorithm 1), applied to one image of the *Actinomyces israeli* class.

4.3.1 EfficientNet

It was published under the name *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks* [35]. One of the main contributions of the paper was a scaling method that uniformly scales all dimensions of a network (depth, width, and resolution), by using a *compound coefficient*. The compound scaling method uses a compound coefficient ϕ to uniform scale the network dimensions. Depth (d), width (w), and resolution (r) of the network can be calculated with ϕ as follows:

$$\begin{aligned} d &= \alpha^{\phi}, \\ w &= \beta^{\phi}, \\ r &= \gamma^{\phi}, \end{aligned} \tag{1}$$

where α, β, and γ are constants that can be determined by a small grid search [35].

The authors proposed a family of models, called *EfficientNets*. The largest model achieved a state-of-the-art score of 84.3% of top-1 accuracy on ImageNet classification while being 8.4x smaller and 6.1x faster on inference than the best existing ConvNet at that moment [35]. Their base architecture is named EfficientNet-B0 and contains a total of 5.3 million parameters. EfficientNet-B1 to B7 were obtained by using different ϕ values on equation 1.

We used the implementation and pre-trained models of EfficientNet in PyTorch, provided by *lukemelas* [2]. The experiments include the three lighter versions of the EfficientNet, i.e., all the versions with less than 10 million parameters. The architectures are named as follows:

— efficientnet-b0: EfficientNet-B0, initialized with a pre-trained model on ImageNet.

— efficientnet-b1: EfficientNet-B1, initialized with a pre-trained model on ImageNet.

— efficientnet-b2: EfficientNet-B2, initialized with a pre-trained model on ImageNet.

[2] https://github.com/lukemelas/EfficientNet-PyTorch#loading-pretrained-models

All the EfficientNet-based architectures were trained using transfer learning. The original B0, B1, and B2 versions had a total of 5.3, 7.8, and 9.2 million parameters. We performed fine-tuning by modifying the fully connected layer of the network to have 32 output neurons. the final architectures have a total of 4, 6.6, and 7.7 million parameters, respectively (details in Table **??**).

4.3.2 MobileNet V2

MobileNet V2 was first reported in the paper *MobileNetV2: Inverted Residuals and Linear Bottlenecks* [27], where the authors proposed a new mobile architecture, based on their main contribution: a novel layer, called inverted residual with linear bottleneck. This novel layer takes a low-dimensional compressed representation as input, and then it is expanded to a high dimension, where it is filtered with a lightweight depth-wise convolution. The base architecture had a total of 3.4 million parameters and achieved a top-1 accuracy score of 72.0% over the ImageNet classification.

In our experiments, we used the official PyTorch implementation of the MobileNetV2, as well as its pre-trained model on ImageNet. We present just one version of this architecture, which is called mobilenet_v2.

We used transfer learning to initialize our architecture; for fine-tuning, we changed the network topology by modifying the last fully connected layer of the original MobileNetV2. In the end, the network had a total of 32 output neurons. After fine-tuning, the total parameter number of the network was reduced to 2.2 millions.

4.3.3 MobileNet V3

Andrew Howard et al, presented the next generation of MobileNets (direct sucessor of MobileNetV2) in the paper *Searching for MobileNetV3* in 2019 [9]. They tuned MobileNetV3 to be efficient in mobile CPUs with a combination of hardware-aware network architecture search (NAS) and complementary approaches. The authors released two versions: MobileNetV3-large and MobileNetV3-small, with 5.4 and 2.5 million parameters, respectively, and claimed that, compared to MobileNetV2, the largest model is 3.2% more accurate on ImageNet classification while reducing latency by 15%. In top-1 classification accuracy of ImageNet, the "-large" model achieved 73.8% and the "-small" model achieved a 64.9%.

Our experiments were performed with a GitHub implementation of MobileNetV3 [3]. Code and pre-trained models come from the same source. We used both the large and small version of the network to generate our solutions, which are named as follows:

— mobilenet_v3_small: Implemented and trained as described in the original paper.

— mobilenet_v3_large: Implemented and trained as described in the original paper.

As with the other architectures, we used transfer-learning to initialize the networks, then, we fine-tuned each version by modifiying the last fully connected layer, in a way that the output is composed by 32 neurons. After fine-tuning, the number of parameters dropped from 5.4 and 2.5 millions, to 1.5 and 4.2 millions, for the large and small models, respectively.

4.3.4 ShuffleNet V2

First presented in the paper *ShuffleNet V2: Practical Guidelines for Efficient CNN Architecture Design* [17]. Their best proposal achieved a top-1 accuracy score of 73.3% on ImageNet classification.

All the experiments were carried out using the official PyTorch [22] implementation of this architecture. We used pre-trained models when available on the PyTorch model zoo. The main difference among the four implementations lies in the output channel multiplier, which can be 0.5, 1, 1.5, or 2, which leads the number of parameters of the network to be 1.4, 2.3, 3.5, and 7.4 millions, respectively.

In total, we present experiments with 4 different ShuffleNet V2-based architectures, which are named as follows:

[3] https://github.com/d-li14/mobilenetv3.pytorch

— shufflenet_v2_x0_5: ShuffleNet V2 with a 0.5 output channel multiplier. This version was initialized with a pre-trained model on ImageNet.

— shufflenet_v2_x1_0: ShuffleNet V2 with a 1.0 output channel multiplier. This version was initialized with a pre-trained model on ImageNet.

— shufflenet_v2_x1_5: ShuffleNet V2 with a 1.5 output channel multiplier. This version was trained from scratch because no pre-trained models were available.

— shufflenet_v2_x2_0: ShuffleNet V2 with a 2.0 output channel multiplier. This version was trained from scratch because no pre-trained models were available.

As we said before, we used transfer-learning in shufflenet_v2_x0_5 and shufflenet_v2_x1_0; the remaining architectures were trained from scratch. For fine-tuning, we changed the network topology by modifying the last fully connected layer of all the architectures in a way that all of them had a total of 32 output neurons (which is the number of classes in our problem). After fine-tuning, the number of parameters of the networks with output channel multipliers of 0.5, 1, 1.5, and 2 was reduced to 0.3, 1.2, 2.5 and 5.5 millions, respectively.

4.3.5 SqueezeNet

It appeared for the first time in the paper *SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters and <0.5 MB Model Size* [10], as a proposal to provide some of the advantages of small CNNs, such as: less communication across servers during distributed training, less bandwith to export models from the cloud to autonomous cars and to provide CNNs feasible to deploy on field-programmable gate arrays (FPGAs) and other hardware with limited memory. Their base architecture achieved a top-1 accuracy score of 57.5% on the ImageNet classification task. Their main contribution consisted in a model size reduction up to 510x, when comparing to AlexNet [14]; also, SqueezeNet models can be saved to disk with a size between 0.47 and 4.8 MB. The vanilla version had a total of 1.248 million parameters.

Similar to MobileNetV2 and ShuffleNetV2, for SqueezeNet we used the official PyTorch implementation and the pre-trained models on ImageNet, all available on the PyTorch model zoo.

We present two SqueezeNet-based architectures to solve our problem, which are called as follows:

— squeezenet1_0: Vanilla version of SqueezeNet, also called SqueezeNet 1.0. This version was initialized with a pre-trained model on ImageNet.

— squeezenet1_1: A slightly modified version of SqueezeNet, also called SqueezeNet 1.1. This version was initialized with a pre-trained model on ImageNet. SqueezeNet 1.1 has 2.4x less computation and 12,000 less parameters.

After transfer-learning, we fine-tuned both networks by modifying the last convolutional layer, to have a total of 32 output filters in the last convolutional layer. After fine-tuning, the total parameters of the network reduced to 0.751 and 0.738 millions in squeezenet1_0 and squeezenet1_1, respectively.

4.4 Experiments

Above we have just described our architectures and some details about the implementation, in this section we are providing details about the process of training and evaluation with cross-validation.

4.4.1 Loss function, optimizer and hyper-parameters

The following list contains relevant information about our experimental setup:

— All architectures were trained by using the cross-entropy loss function (a combination between log softmax and the negative log likelihood loss),

— All architectures use the Adam optimizer [13], with the L2 penalty implemented as proposed in the paper *Decoupled Weight Decay Regularization* [16],

— All architectures are trained end-to-end, even when we used transfer-learning to initialize the weights, this allows the models to learn more problem-specific details,

— Learning rate, weight decay and the number of epochs are adjusted according to the performance of each architecture when training,

— Our cross-validation setup uses 10 folds in everyone of our experiments, the dataset is randomized but uses the same random state in all experiments (to ensure uniform and fair evaluations),

4.4.2 Software

The following list contains relevant information about the language, libraries and software:

— Dataset scripts, training and evaluation code are fully wrote in Python 3,

— We used PyTorch [23] as our main framework,

— The process of training and evaluation of each architecture is a Jupyter Notebook [4], in total, we have two notebooks for each architecture, one for the original dataset and one for the augmented version,

— All models are evaluated with five metrics: top-1 and top-5 accuracy, weighted precision, weighted recall and weighted F1 score. For the last three, we used the Scikit-Learn implementation [24].

[4] https://jupyter.org/

4.4.3 Hardware

Our main workstation during experiments works with an Intel i7-9750H processor, 20 gigabytes of RAM, and a commercial NVIDIA GTX 1660-Ti with 6 gigabytes of VRAM. All experiments, in training and evaluation phases fitted in just 6 gigabytes of VRAM and less of 10 gigabytes of RAM, ensuring the repeatability of our experiments without needing specific or high-tier hardware.

5 Results

In order to present a complete comparison, we provide three different tables:

— Table 2 contains the results of cross-validation with all our proposals, using the original dataset,

— Table 3 contains the results of cross-validation with all our proposals, using our augmented version of the dataset,

— Table **??** shows a comparison of the number of parameters and Top-1 accuracy among literature methods and our methods.

All values in the three tables are averages of the 10 folds.

6 Discussion

7 Conclusions

References

1. **Abd Elaziz, M., Hosny, K. M., Hemedan, A. A., & Darwish, M. M. (2020).** Improved recognition of bacterial species using novel fractional-order orthogonal descriptors. *Applied Soft Computing*, Vol. 95, pp. 106504.
2. **Ahmed, W. M., Bayraktar, B., Bhunia, A. K., Hirleman, E. D., Robinson, J. P., & Rajwa, B. (2012).** Classification of bacterial contamination using image processing and distributed computing. *IEEE journal of biomedical and health informatics*, Vol. 17, No. 1, pp. 232–239.

Method	# Parameters	Total samples	# Folds	# Epochs per fold	Top-1 Accuracy	Top-5 Accuracy	Precision	Recall	F1 Score
shufflenet_v2_x0_5	0.374 M	669	10	10	0.8893	**0.9954**	0.9125	0.8893	0.8810
squeezenet1_0	0.751 M	669	10	15	0.4487	0.8895	0.4366	0.4487	0.4047
squeezenet1_1	0.738 M	669	10	15	0.5810	0.9686	0.5675	0.5810	0.5403
shufflenet_v2_x1_0	1.286 M	669	10	10	0.9207	0.9940	0.9412	0.9207	0.9170
mobilenet_v3_small	1.505 M	669	10	10	**0.9341**	**0.9954**	**0.9548**	**0.9341**	**0.9342**
mobilenet_v2	2.264 M	669	10	10	0.9237	0.9910	0.9488	0.9237	0.9236
† shufflenet_v2_x1_5	2.511 M	669	10	20	0.6307	0.9626	0.6846	0.6307	0.6111
Efficientnet-b0	4.048 M	669	10	15	0.8938	0.9895	0.9072	0.8938	0.8873
mobilenet_v3_large	4.243 M	669	10	10	0.9132	0.9925	0.9311	0.9132	0.9077
† shufflenet_v2_x2_0	5.543 M	669	10	20	0.6486	0.9655	0.6827	0.6486	0.6266
efficientnet-b1	6.654 M	669	10	10	0.8354	0.9835	0.8579	0.8354	0.8224
efficientnet-b2	7.746 M	669	10	10	0.8668	0.9940	0.8837	0.8668	0.8558

Table 2. Cross-validation results of our methods when using the original version of the DIBaS dataset. Models with a † were trained from scratch due to a lack of pre-trained models, the remaining ones were initialized with transfer-learning.

Method	# Parameters	Total samples	# Folds	# Epochs per fold	Top-1 Accuracy	Top-5 Accuracy	Precision	Recall	F1 Score
shufflenet_v2_x0_5	0.374 M	24073	10	10	0.9557	0.9976	0.9602	0.9570	0.9571
squeezenet1_1	0.738 M	24073	10	15	0.9136	0.9955	0.9210	0.9136	0.9125
squeezenet1_0	0.751 M	24073	10	15	0.8882	0.9937	0.9007	0.8882	0.8866
shufflenet_v2_x1_0	1.286 M	24073	10	10	0.9635	0.9978	0.9666	0.9635	0.9636
mobilenet_v3_small	1.505 M	24073	10	10	0.9702	0.9975	0.9711	0.9702	0.9703
mobilenet_v2	2.264 M	24073	10	15	0.9737	0.9980	0.9745	0.9737	0.9737
† shufflenet_v2_x1_5	2.511 M	24073	10	15	0.9001	0.9906	0.9073	0.9001	0.9000
Efficientnet-b0	4.048 M	24073	10	10	0.9720	0.9976	0.9728	0.9720	0.9720
mobilenet_v3_large	4.243 M	24073	10	10	0.9717	0.9978	0.9729	0.9717	0.9718
† shufflenet_v2_x2_0	5.543 M	24073	10	15	0.8984	0.9972	0.9706	0.8984	0.8977
efficientnet-b1	6.654 M	24073	10	10	0.9661	0.9965	0.9676	0.9661	0.9661
efficientnet-b2	7.746 M	24073	10	10	0.9715	0.9970	0.9723	0.9715	0.9715

Table 3. Cross-validation results of our methods when using the augmented version of the DIBaS dataset. Models with a † were trained from scratch due to a lack of pre-trained models, the remaining ones were initialized with transfer-learning.

3. **Aslam, B., Basit, M., Nisar, M. A., Khurshid, M., & Rasool, M. H. (2017).** Proteomics: technologies and their applications. *Journal of chromatographic science*, Vol. 55, No. 2, pp. 182–196.

4. **Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009).** Imagenet: A large-scale hierarchical image database. *2009 IEEE conference on computer vision and pattern recognition*, Ieee, pp. 248–255.

5. **Forero, M. G., Cristóbal, G., & Desco, M. (2006).** Automatic identification of mycobacterium tuberculosis by gaussian mixture models. *Journal of microscopy*, Vol. 223, No. 2, pp. 120–132.

6. **Franco-Duarte, R., Černáková, L., Kadam, S., S Kaushik, K., Salehi, B., Bevilacqua, A., Corbo, M. R., Antolak, H., Dybka-Stępień, K., Leszczewicz, M., et al. (2019).** Advances in chemical and biological methods to identify microorganisms—from past to present. *Microorganisms*, Vol. 7, No. 5, pp. 130.

7. **Gallardo-García, R., Jarquín-Rodríguez, A., Beltrán-Martínez, B., & Martínez, R. (2020).** Deep learning for fast identification of bacterial strains in resource constrained devices. *X Congreso Nacional y II Congreso Internacional de Ciencias de la Computación*, pp. .

8. **Holmberg, M., Gustafsson, F., Hörnsten, E. G., Winquist, F., Nilsson, L. E., Ljung, L., & Lundström, I. (1998).** Bacteria classification based on feature extraction from sensor data. *Biotechnology techniques*, Vol. 12, No. 4, pp. 319–324.

9. **Howard, A., Sandler, M., Chu, G., Chen, L.-C., Chen, B., Tan, M., Wang, W., Zhu, Y., Pang, R., Vasudevan, V., et al. (2019).** Searching for mobilenetv3. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1314–1324.

10. **Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016).** Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. *arXiv preprint arXiv:1602.07360*.

11. **Jenkins, S. A., Drucker, D., Keaney, M. G., & Ganguli, L. A. (1991).** Evaluation of the rapid id 32a system for the identification of bacteroides fragilis and related organisms. *Journal of applied bacteriology*, Vol. 71, No. 4, pp. 360–365.

12. **Khalifa, N. E. M., Taha, M. H. N., Hassanien, A. E., & Hemedan, A. A. (2019).** Deep bacteria: robust deep learning data augmentation design for limited bacterial colony dataset. *International Journal of Reasoning-based Intelligent Systems*, Vol. 11, No. 3, pp. 256–264.

13. **Kingma, D. P. & Ba, J. (2014).** Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

14. **Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012).** Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, Vol. 25, pp. 1097–1105.

15. **Liu, J., Dazzo, F. B., Glagoleva, O., Yu, B., & Jain, A. K. (2001).** Cmeias: a computer-aided system for the image analysis of bacterial morphotypes in microbial communities. *Microbial Ecology*, Vol. 41, No. 3, pp. 173–194.

16. **Loshchilov, I. & Hutter, F. (2017).** Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

17. **Ma, N., Zhang, X., Zheng, H.-T., & Sun, J. (2018).** Shufflenet v2: Practical guidelines for efficient cnn architecture design. *Proceedings of the European conference on computer vision (ECCV)*, pp. 116–131.

18. **MedlinePlus (2020).** Bacterial infections.

19. **Mishra, M. & Chauhan, P. (2016).** Applications of microscopy in bacteriology. *Microscopy Research*, Vol. 4, No. 1, pp. 1–9.

20. **Nasip, Ö. F. & Zengin, K. (2018).** Deep learning based bacteria classification. *2018 2nd international symposium on multidisciplinary studies and innovative technologies (ISMSIT)*, IEEE, pp. 1–5.

21. **Nobile, C. J. & Johnson, A. D. (2015).** Candida albicans biofilms and human disease. *Annual review of microbiology*, Vol. 69, pp. 71–92.

22. **Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017).** Automatic differentiation in pytorch.

23. **Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019).** Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, pp. 8026–8037.

24. **Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D.,**

Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, Vol. 12, pp. 2825–2830.

25. **Perronnin, F. & Dance, C. (2007).** Fisher kernels on visual vocabularies for image categorization. *2007 IEEE conference on computer vision and pattern recognition*, IEEE, pp. 1–8.
26. **Rujichan, C., Vongserewattana, N., & Phasukkit, P. (2019).** Bacteria classification using image processing and deep convolutional neural network. *2019 12th Biomedical Engineering International Conference (BMEiCON)*, IEEE, pp. 1–4.
27. **Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018).** Mobilenetv2: Inverted residuals and linear bottlenecks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4510–4520.
28. **Satoto, B. D., Utoyo, M. I., Rulaningtyas, R., & Koendhori, E. B. (2020).** An auto contrast custom convolutional neural network to identifying gram-negative bacteria. *2020 International Conference on Computer Engineering, Network, and Intelligent Multimedia (CENIM)*, IEEE, pp. 70–75.
29. **Sauer, S. & Kliem, M. (2010).** Mass spectrometry tools for the classification and identification of bacteria. *Nature Reviews Microbiology*, Vol. 8, No. 1, pp. 74–82.
30. **Simonyan, K. & Zisserman, A. (2014).** Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
31. **Singhal, N., Kumar, M., Kanaujia, P. K., & Virdi, J. S. (2015).** Maldi-tof mass spectrometry: an emerging technology for microbial identification and diagnosis. *Frontiers in microbiology*, Vol. 6, pp. 791.
32. **Sommer, C. & Gerlich, D. W. (2013).** Machine learning in cell biology–teaching computers to recognize phenotypes. *Journal of cell science*, Vol. 126, No. 24, pp. 5529–5539.
33. **Spratt, D. A. (2004).** Significance of bacterial identification by molecular biology methods. *Endodontic topics*, Vol. 9, No. 1, pp. 5–14.
34. **Talo, M. (2019).** An automated deep learning approach for bacterial image classification. *arXiv preprint arXiv:1912.08765*.
35. **Tan, M. & Le, Q. V. (2019).** Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*.
36. **Trattner, S., Greenspan, H., Tepper, G., & Abboud, S. (2004).** Automatic identification of bacterial types using statistical imaging methods. *IEEE transactions on medical imaging*, Vol. 23, No. 7, pp. 807–820.
37. **Váradi, L., Luo, J. L., Hibbs, D. E., Perry, J. D., Anderson, R. J., Orenga, S., & Groundwater, P. W. (2017).** Methods for the detection and identification of pathogenic bacteria: past, present, and future. *Chemical Society Reviews*, Vol. 46, No. 16, pp. 4818–4832.
38. **Zieliński, B., Plichta, A., Misztal, K., Spurek, P., Brzychczy-Włoch, M., & Ochońska, D. (2017).** Deep learning approach to bacterial colony classification. *PloS one*, Vol. 12, No. 9, pp. e0184554.