applied sciences

MDPI

Review

Analysis of Machine Learning Techniques for Information Classification in Mobile Applications

Sandra Pérez Arteaga, Ana Lucila Sandoval Orozco and Luis Javier García Villalba *

Group of Analysis, Security and Systems (GASS), Department of Software Engineering and Artificial Intelligence (DISIA), Faculty of Computer Science and Engineering, Office 431, Universidad Complutense de Madrid (UCM), Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, Spain
* Correspondence: javiergv@fdi.ucm.es; Tel.: +34-91-394-7638

Abstract: Due to the daily use of mobile technologies, we live in constant connection with the world through the Internet. Technological innovations in smart devices have allowed us to carry out everyday activities such as communicating, working, studying or using them as a means of entertainment, which has led to smartphones displacing computers as the most important device connected to the Internet today, causing users to demand smarter applications or functionalities that allow them to meet their needs. Artificial intelligence has been a major innovation in information technology that is transforming the way users use smart devices. Using applications that make use of artificial intelligence has revolutionised our lives, from making predictions of possible words based on typing in a text box, to being able to unlock devices through pattern recognition. However, these technologies face problems such as overheating and battery drain due to high resource consumption, low computational capacity, memory limitations, etc. This paper reviews the most important artificial intelligence algorithms for mobile devices, emphasising the challenges and problems that can arise when implementing these technologies in low-resource devices.

Keywords: algorithms; architectures; artificial intelligence; challenges; classification; deep learning; federated learning; limited resources; mobile devices

Citation: Pérez Arteaga, S.; Sandoval Orozco, A.L.; García Villalba, L.J. Analysis of Machine Learning Techniques for Information Classification in Mobile Applications. *Appl. Sci.* **2023**, *13*, 5438. https://doi.org/10.3390/app13095438

Academic Editor: Yu-Dong Zhang

Received: 11 February 2023
Revised: 13 April 2023
Accepted: 23 April 2023
Published: 27 April 2023

1. Introduction

Smartphones have become an essential part of our daily lives and are considered personal devices for individuals. Initially, smartphones were primarily intended for business [1]; however, in a few years, they have replaced computers as the most important device connected to the Internet [2], leading to a high demand for smarter applications or functionalities to meet consumer needs.

Artificial intelligence has been a topic of controversy in all areas since its implementation in computer video games. This technology has opened to many fields of knowledge, making a solid path for smartphones to perform many tasks that make people's lives easier, such as recognising places through a photograph taken from the device's camera, interpreting voice commands, biometric pattern recognition to unlocking devices, automatic emergency calls when a medical mishap occurs, etc. In addition, artificial intelligence can strike the right balance between hardware and software, allowing for more sophisticated functionalities. Therefore, it is possible to process, analyse and implement the optimal configuration of resources to increase the lifetime of devices and balance the hardware features of cameras to take pictures in extreme lighting conditions or, on the contrary, in low light for night photography. In addition, artificial intelligence has been part of many devices in recent years such as so-called smart homes, which can be managed automatically via a mobile app. Companies such as Google and Amazon have created innovation in common devices that we use in our homes, such as lamps, speakers, appliances, etc., as well as smart home devices that help people with everyday tasks, with products such as Alexa

or Google Assistant. However, with the constant demand for new functionalities, the currently used artificial intelligence algorithms are becoming obsolete, and as improvements are made, their computational cost and demand for resources increases, which means that optimisation methods must be sought to be implemented in low-resource devices.

This paper presents a review of the main artificial intelligence algorithms that can be adapted for mobile devices to fulfill image and text classification functionalities, as well as providing a comparison of the main algorithms and methods analysed. The rest of this paper is organised as follows. Section 2 describes the challenges and problems that may arise when implementing machine learning in mobile devices. Section 3 details the architectures that can be used to implement machine learning on mobile devices. Section 4 describes the algorithms that can be implemented for image and video detection. Section 5 describes the algorithms that are used for text analysis on mobile devices. Section 6 describes the different frameworks available to implement machine learning on mobile devices. Section 7 mentions how federated learning works, and finally, Section 8 presents the conclusions of the work.

2. Challenges and Issues

The use of artificial intelligence algorithms in mobile devices has helped to maximise the functionalities we can perform with a mobile phone in different areas of research; processing input data, such as images, text or audio; image recognition; object detection; and gesture recognition, taking into account the CPU (central processing unit) and GPU (graphics processing unit) of the device.

Some advantages of machine learning on mobile devices are low latency, privacy, offline operation and low or no cost. However, mobile devices can be restricted in terms of storage, memory, computing resources and power consumption [3].

Challenges that can arise with the implementation of mobile algorithms include the conversion of large models to device-friendly models such as CoreML, TFLite, Edge ML or WinML using, for example, Tensorflow with a performance advantage. Other challenges that may arise are associated the version of the mobile device, as not all have updates or will soon be discontinued, as well as the version of the SDK [4].

Common challenges include limited computational and memory power when running machine learning processes consuming a lot of device resources. This also includes issues of limited bandwidth and connectivity, as well as easy and efficient deployment of the models, as they have to be customized and adapted to their intended use, taking into account device deployment management, as well the quality of the data with which the models are trained and data overfitting, whereby the model becomes adjusted to the training data to such an extent that it does not generalize well to the test data [5,6].

It is also necessary to take into account the security of the data that can be used in the models or the data that are processed on mobile devices, since the information may contain sensitive data that could damage the integrity of users.

3. Architectures for Machine Learning on Mobile Devices

Machine learning models consist of two fundamental phases that can be performed in cloud or on-device architectures: training and inference. The training of a model involves finding patterns and grouping them according to their similarity, with the aim of minimising losses in most test examples; this process is called empirical risk minimisation. Inference is the process whereby the trained model is tested; once the AI learns the patterns, it creates an inference model that it uses to solve or classify a given problem. This base architecture allows AI models to learn complex structures without requiring large amounts of data. In this section, we review architectures that are used to implement machine learning models on mobile devices, emphasising the advantages and disadvantages of their use, such as cloud, on-device and hybrid architectures.

3.1. Cloud

For the development of machine learning functions, the type of development to be carried out must be taken into account, as this will determine how functional the proposal is and how well it will meet the needs of the target audience. There are two types of paradigms associated with the infrastructure that makes use of cloud services for machine learning. The first involves the use of cloud computing only to make inferences, and the second involves the developer performing the training and testing using cloud infrastructure, as explained below.

3.1.1. Testing without Training

Currently, there is a large number of applications that make use of platforms dedicated to providing machine learning services by performing their functions in the cloud. These platforms are called machine learning as a service (MLaaS) [7]. The MLaaS concept is an umbrella term for various cloud-based platforms that use machine learning to provide functionalities such as predictive analytics, data preprocessing, model creation, execution orchestration, model deployment, etc. Companies such as Google, IBM, Microsoft, Oracle and Amazon, among others, offer such services. Table 1 shows platforms that can provide applications created by developers with specialised servers to deploy complex, resource-intensive machine learning functions [8–12]. Generally, the models on these platforms are retrained on a regular basis, supporting high confidence in the predictions.

This architecture is the most beneficial for applications that are mounted on devices with very low performance, as it allows only a small amount of data to be sent to the server that performs all the processing work; however, this architecture does not ensure privacy, as the data used to make inferences leave the device. This problem increases when the data that travel are personal, as any vulnerability in the APIs could compromise access to these data.

Table 1. Cloud Machine Learning comparison.

Service	Description	Interface	Models	Extras
Amazon [12]	Automated infrastructure that applies ML techniques to information stored on Amazon Web Services.	Amazon ML console, Amazon CLI	Users can use their information with pretrained algorithms that can be included in: - Regression; - Binary classification; - Multiclass classification.	Additional payments for information stored in a collection of cloud computing services billed separately.
Google Cloud [8]	Gives customers access to cutting-edge algorithms used by Google with the help of other industry-leading applications for use in searches. Users have the ability to make their own algorithms.	The terminal is run using gcloud ml-engine to control tensor flow processes.	Customers have the ability to create models or use pretrained models that are supported by following apps: -Multimedia analysis (image and video); -Dialogue recognition; -Text analysis; -Translation.	Google account required.

Table 1. *Cont.*

Service	Description	Interface	Models	Extras
IBM Watson [9]	Focuses on putting algorithms into production using REST API connectors.	-IBM's SPSS graphical analysis software can be used as a front end; -API connectors allow customers to design models in third-party data science applications.	Users can design algorithms in any language using REST API connectors. Access to Apache Speak's MLlib library of machine learning models is available through IBM's Data Science Experience workbench platform (implementation currently in a closed beta).	A Bluemix account is required.
Microsoft Azure [10]	Includes predefined models that clients can use on their data.	Azure Machine Learning Studio, R and Python coding.	Customers may use their information in algorithms, including: - Decision tree; -Bayesian systems; -Deep neural networks; - Decision jungles; -The rating service supports these algorithms; -Binary classification. -Regression clustering.	A paid Azure account and a free Microsoft account are required.
Oracle [11]	Oracle is a database architecture relational in which data are managed and processed over local and wide area networks. The Oracle database has its own networking component to enable communications across networks.	Oracle machine learning AutoML.	Machine learning function.	Oracle Platform account required.

Figure 1 shows a general outline of a basic machine learning architecture. First, an app makes a request for information to the API about the image, the API communicates with the application through the remote server and data are sent to the cloud. Then, the platform makes a prediction about the image and returns the result of the prediction. This architecture is popular among service applications to define usage trends such as music preferences based on played tracks or for video streaming services to generate recommendations based on tastes and content viewing time.

3.1.2. Training and Testing

One of the biggest limitations of mobile devices in the implementation of machine learning models in their functions is that when training and testing are performed, a large amount of resources such as RAM memory, energy and time is consumed. This is because in order for the models to be highly accurate, large datasets are necessary for the model to learn the patterns and make more accurate inferences. Therefore, in some situations,

these two functions must be performed in a cloud architecture as in [13–19]. A summary of research using a cloud approach for data training and inference is provided in Table 2.

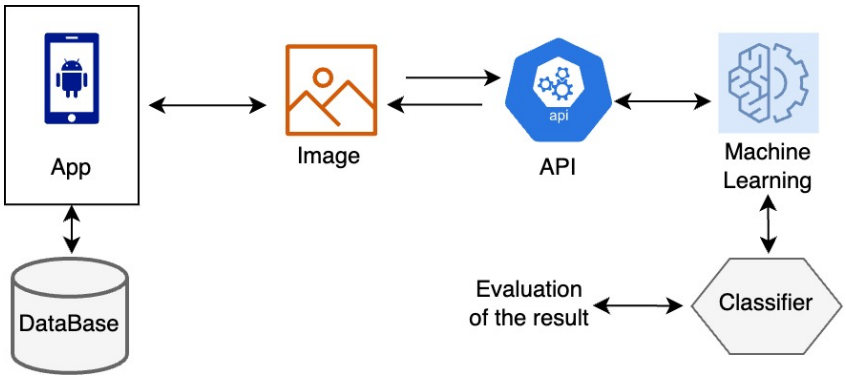

Figure 1. Machine learning online model architecture.

Table 2. Overview of proposals for cloud inference and training.

Approach	Architecture	Device or Technology	Proposal Scope	Reference
JoinDNN	Hybrid	Mobile	Computing with a mobile device and the cloud	Eshratifar et al. [13]
Pie-NET	Cloud	3D Points	Parametric inference of edges	Wang et al. [14]
MEANet	Cloud	IoT	Image Classification	Long et al. [15]
SPINN	Cloud	CNN	CNN splitting at run time	Laskaridis et al. [16]
DATAMIX	Hybrid	Edge devices	Speech recognition	Liu et al. [17]
PieSlicer	Cloud	Online services	Cloud-based CNN inference	Ogden et al. [18]
Deep Learning Inference on Real-time	Cloud	DNN	Cloud development	Li et al. [19]

As shown in Figure 2, the architecture does not change with respect to the previous one; the only difference is that the service provider enables model training with our own datasets, which leads to a design more tailored to the needs of our application and customised to our data. The developer can upload the data for training either from the application itself or using a different service and use them to perform the training in the cloud. However, this scheme has the same privacy issues as the previous one.

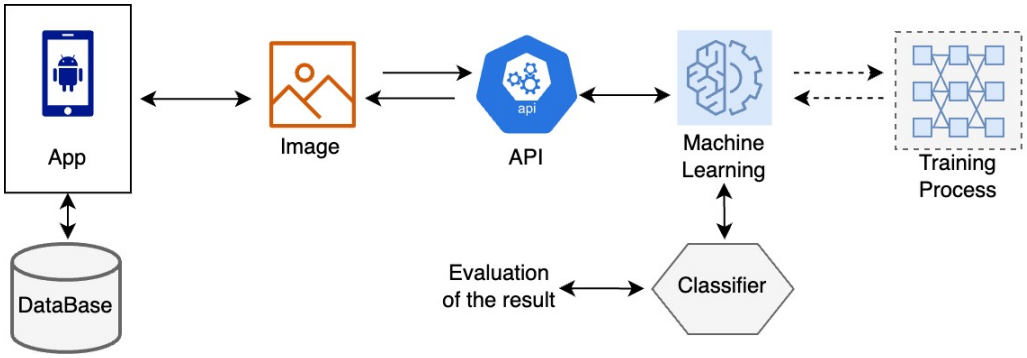

Figure 2. Machine learning online model architecture: training and testing in the cloud.

3.2. On-Device Architecture

One of the architectures that is rarely used involves training and inference on the device because machine learning algorithms are often not optimized for low-resource

devices, which leads to a high consumption of power and resources and high latency. This architecture helps maintain user privacy, as user data remain on the device.

3.2.1. On-Device Testing with Pretrained Models

Making predictions on the device has its advantages. First, privacy is increased because the data do not leave the device at any time, and there is no need for an API for the application to communicate with the machine learning model, which helps to avoid the introduction of an element into the architecture that could put the user's data at risk. Secondly, it lowers the response time from the time the request is made to the model until a prediction is made, achieving latencies of the order of microseconds in particular cases, as well as improving the amount of bandwidth used. This low latency is fundamental for architectures for which response time is crucial, such as an autonomous car. In addition, because device models must be as small as possible, it affects the ability of inference accuracy and flexibility of use, as they are tailored to specific circumstances.

In order to run machine learning models on a limited-resource device, it is necessary to convert the original model files into TensorFlow Lite files. This is a framework specially designed for running deep learning models on the device, storing models in a special low-storage file format that enables low execution rates by reducing computing and memory requirements [20]. As seen in Figure 3, to perform inference on a given image, the application must load the pretrained model from internal storage and then perform the necessary computations locally on the device. Compared to previous architectures, this one does not need to communicate with a remote server or API to perform its functions. The developer only needs to check if the pretrained model meets the application requirements, then load it into the application. If not, a custom model is needed, which can be trained and adjusted to the application's needs on a computer or in the cloud. The pretrained model can be a common model used for a specific purpose or tailored to research needs, such as in [21–26].

As previously discussed, an application that performs all data processing on the device using algorithms preloaded into the device's storage is appropriate when it is necessary to preserve the user's privacy, as the user's data never leave the device. An example is the solution presented in [27], in which the authors presented a solution for detection of spam in short text messages (SMS), as such messages may contain sensitive user information that, if disclosed, could compromise some aspect of the user's personal life.

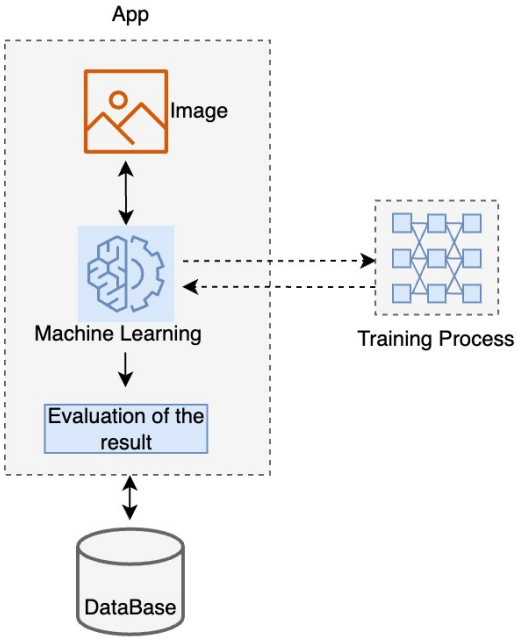

Figure 3. Machine learning model architecture on-device testing.

3.2.2. Training and Testing on the Device

This architecture is perhaps the most optimal for low-scale data processing (Figure 4). It preserves user privacy, as training and inferences are performed on the device, allowing the application to continue learning from the data first hand. In addition, it helps to avoid the costs and bandwidth requirement associated with using cloud services. This architecture is feasible in scenarios using smaller machine learning algorithms. As shown above, using pretrained models stored on the device to make predictions leads to a decrease in the accuracy of the results. This is mainly due to the use of optimised models that are too small or basic ML algorithms, given the processing limitations that may be encountered, as in [24].

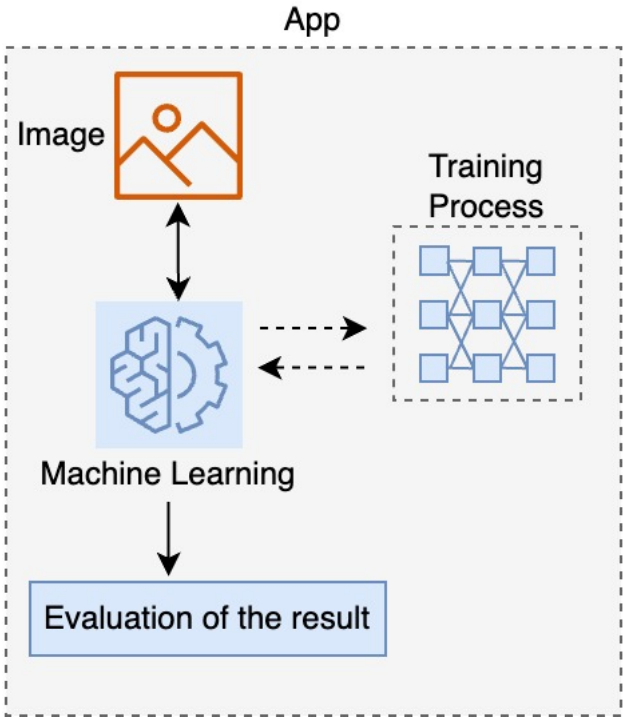

Figure 4. Machine learning model architecture on-device testing and inference.

3.3. Hybrid

This architecture involves two stages, the first of which occurs on the mobile device and the second of which occurs in the cloud. In a hybrid architecture of the automatic learning model, the model performs the extraction of internal characteristics; these characteristics feed the models of the second stage to predict the response variable [28]. This architecture allows the model to be refined by adapting it to the individual model and customising it to the user's individual data.

4. Image and Video Classification Algorithms on Mobile Devices

The algorithms that are used for image classification are mostly used for computers, which leads to more computational complexity and processing power for training data, which, when performing image processing such as image classification to identify a person in a photo on a mobile device, is done with algorithms with low computational cost to avoid latency, high memory consumption, high battery consumption and other potential problems. Currently, deep learning models are often used for image classification and recognition; the main algorithms used for this type of classification in mobile devices are described in the next sections.

4.1. MobileNet

MobileNet is a model that can be deployed on mobile devices to meet the intelligence needs of the market for mobile applications due to the low number of parameters required for training, low latency in processing inferences and low consumption of computing resources. Mobilenet consists of an architecture based on depth-separated convolutions, allowing the model to be quite light and efficient. It can be used for classification, detection, embedding and segmentation and is used in a similar way to other large-scale models [29,30].

- MobileNetV2 is the second version of the MobileNet model that greatly improves the accuracy and inference time of the model. This update contains a full convolution layer with 32 filters and 19 bottleneck layers and direct access connections [29,31].
- MobileNetV3: This version of the model is based on the EfficientNet search method with specific parameter space targets required for use on mobile devices. It is a lightweight model that allows for image classification with low inference times and fits architectures with limited computational resources [32,33].

Among the areas in which this model has been used to improve and automate processes are botany, botanics [34–37], medicine [38–42], manufacturing [43], zoology [44], digital forensics processes [45], etc.

MobileNet Architecture

As discussed above, MobileNet is based on an architecture based on depth-separable convolutions. This convolution consists of two operations: depthwise convolution and pointwise convolution. Depthwise convolution applies separate convolutions to each channel of the input tensor, i.e., a traditional $n \times m$ convolution on a colour image. Subsequently, the activation maps resulting from the convolution operations are concatenated on the depth axis. A traditional 1×1 convolution is then applied to the resulting tensor (pointwise convolution), which combines the channels of the concatenated activation maps [46].

4.2. EfficientNet

Convolutional neural networks (ConvNets) are developed with a fixed resource requirement in mind, then scaled up to obtain better accuracy if more resources are available.

The EfficientNets family of models was designed based on the neural architecture and scaling efficiency of MobileNet and ResNet, achieving better precision and efficiency than their base models. The EfficientNet model, like MobileNet, uses a convolutional neural network (CNN) architecture that improves performance by uniformly balancing and scaling depth, width and resolution using a composite coefficient, limiting those of the convolutional network to fixed portions of the parameters. The composite scaling method intuits that if the input image is larger, the network needs to increase the receptive field by increasing layers and channels to capture detailed patterns from the input image [22].

The EfficientNet model conforms to the MnasNET search method by adding two important concepts, squeeze and excitation blocks (swish activation function and the SE), making use of inventive residual blocks from MobileNetV2 [32].

4.2.1. EfficientNet Architecture

The scaling efficiency of the model is also highly dependent on the reference network. Using the AutoML MNAS framework, a neural architecture is sought to improve the scaling performance in order to optimise the accuracy and efficiency of the model. This architecture makes use of mobile bottleneck inverse convolution (MBConv) techniques such as MobileNetV2 and MnasNet; however, it assumes a larger size due to a higher accuracy and efficiency rate [47], as proposed in [22].

4.2.2. EfficientNet Variants

The EfficientNet model is one of the most relevant among the latest models. This model achieves an assumption of 84.4%. As discussed, EfficientNets models rely on AutoML and composite scaling to increase resource efficiency and achieve superior performance. Following this method, different versions of EfficientNet have been developed; starting from B0, improving the composite scaling method, the EfficientNet B1 to B7 versions were obtained [48]. Although the number of parameters increases, the increase is not significant, while the accuracy increases significantly in contrast to other CNN models [49].

For the generation of new versions of EfficientNet, it is only necessary to scale the network when a set of heuristic scaling characteristics of the base network (B0) is present, which enables the production of increasingly larger networks. In short, each step of a larger network requires a square amount of computation. Therefore, a large amount of training time is necessary to deploy a network that has good accuracy-related results [32]. Table 3 shows the characteristics of each of the EfficientNet-derived models from B0 to B7, as well as the input size in pixels, the number of parameters and the accuracy [22,50].

Table 3. Comparison of features of EfficientNet models.

Version	Input Size (px)	#Params	Accuracy
EfficientNetB0	224 × 224	4,057,253	76.3%/93.2%
EfficientNetB1	240 × 240	6,582,914	78.8%/94.4%
EfficientNet-B2	260 × 260	7,777,012	79.8%/94.9%
EfficientNet-B3	300 × 300	10,792,746	81.1%/95.5%
EfficientNet-B4	380 × 380	17,684,570	82.6%/96.3%
EfficientNet-B5	456 × 456	28,525,810	83.3%/96.7%
EfficientNet-B6	528 × 528	40,973,969	84.0%/96.9%
EfficientNet-B7	600 × 600	64,113,049	84.4%/97.1%

4.2.3. EfficientNet-Lite

EfficientNet-Lite is derived from the state-of-the-art EfficientNet architecture [51]. EfficientNet is a model that has achieved outstanding results in image recognition, speech recognition and video detection and improves the overall prediction/recognition accuracy. This model is suitable for deployment in devices with limited resources, such as mobile phones that implement EfficientNet-Lite, due to low resource consumption and low storage demand due to the low number of parameters needed to train the model. EfficientNet-Lite runs on all mobile CPUs/GPUs/EdgeTPUs [52].

EfficientNet-Lite is based on the efficiency of EfficientNet and is used in edge devices, with five variants that vary in precision and size of the model (number of parameters). EfficientNet-Lite0 is the low-latency and low-size version of the model and, EfficientNet-Lite4 is the high-precision version.

Figure 5 the red line represents the different versions of EfficientNet-lite and the blue line represents popular models for image classification (MobileNetV2, ResNet50 and InceptionV4). The comparison is made in terms of latency and accuracy is quantized with integers only and using the ImageNet dataset running in real time. As shown EfficientNet-Lite4, achieves an accuracy of 80.4% as does the InceptionV4 model, however the latter has a higher latency of approximately 80 ms. This last figure is significant because the processing [53].

Table 4 shows some algorithms that perform well on mobile devices, analysing the input parameters and the output of each of the models in comparison with EfficientNet.

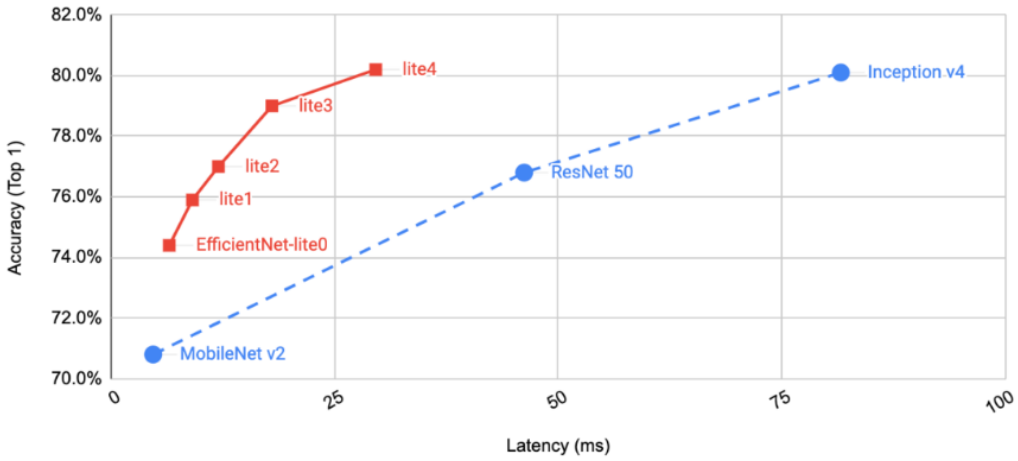

Figure 5. EfficientNet-Lite latency (ms) vs Accuracy (top 1) [53].

Table 4. Characteristics of the operation of models on mobile devices.

PRE-Trained Model	Image Input Size (px)	Acuracy	Parameters	Inference Time	Size
EfficientNetV2 [54]	300 × 300	83%	55 M	57 ms	220 MB
MobileNet [55]	224 × 224	70.4%	4.3 M	22.6s	16 MB
MobileNetV2 [55]	224 × 224	71.3%	3.5 M	25.9 ms	14 MB
ResNet50 [56]	224 × 224	74.9%	25.6 M	58.2 ms	98 MB
VGG16 [57]	224 × 224	71.3%	138.4 M	69.5 ms	528 MB
InceptionV3 [58]	299 × 299	77.9%	23.9 M	42.2 ms	92 MB
NASNetMobile [59]	224 × 224	74.4%	5.3 M	27 ms	23 MB
DenseNEt121 [60]	224 × 224	75%	8.1 M	77.1 ms	33 MB
Xception [59]	299 × 299	79%	22.9 M	109.4 ms	88 MB

The variants of the EfficientNet model shown in Table 3 were used to create the EfficientNet-Lite model, which can be implemented on mobile devices with unlimited resources and perform as well as the EfficientNet model for computers. Table 4 shows algorithms that, thanks to the low number of parameters used for their training, can be adapted to mobile phones; for example, MobileNet and MobileNetV2 achieve a similar accuracy of 70% and 71%, respectively. The number of parameters used is also relatively low, which means that the storage necessary for its operation is around 16 MB. Other algorithms, although they achieve similar performance to MobileNet, require a large amount of resources for processing. A special case is EfficientNet, which despite the higher number of parameters, has versions that can be adapted to a mobile device using TensorflowLite.

5. Text Analytics Algorithms on Mobile Devices

Natural language interface (NLI) helps clients to interact with their computer using high-level language instead of using machine language in the command line interface or with the graphical user interface [61]. NLI involves user–computer interaction, helping the computer to understand high-level language, enabling search queries through text or spoken language.

Mobile devices present usage challenges for users because they are small and have limited resources and network connection, among other elements [62]. Natural language has been implemented in novel applications such as database queries, question answers, personalisation, etc. NLI focuses on the work of desktop computer systems. NLI on mobile devices expands knowledge theoretically and practically to improve mobile devices.

One area that has attracting attention from researchers is the use of lightweight, pretrained deep learning models to perform text processing on devices, which can allow for the classification of text messages, application messages, notifications, etc., without the need to connect to a remote server to make inferences. However, this task is complex due to hardware limitations such as RAM, storage or battery requirements. Algorithms designed for mobiles must have few training parameters to make the model light enough to be supported.

In [63] the authors proposed TinyBERT, a BERT-based model that, to enable its use in resource-constrained devices, makes use of novel methods to distill transformer knowledge both in the pretraining stage and in the task-specific learning stage to be implemented.

On the other hand, in [64], MobileBERT, an algorithm based on BERT_Large designed between self-attenuation and feed-forward networks, was proposed. The results obtained by the authors show that MobileBert is 4.3 times smaller and 5.5 times faster than BERT_Base.

MobileBERT is a bidirectional transformer based on the BERT model, which is compressed and accelerated using various approaches. Masquerade language modeling (MLM) is effective in predicting masquerade tokens and in NLU in general. MLM is not optimal for text generation, but models trained with a causal language modeling (CLM) goal are better in that regard [63]. The architecture of MobileBERT is shown in the Figure 6.

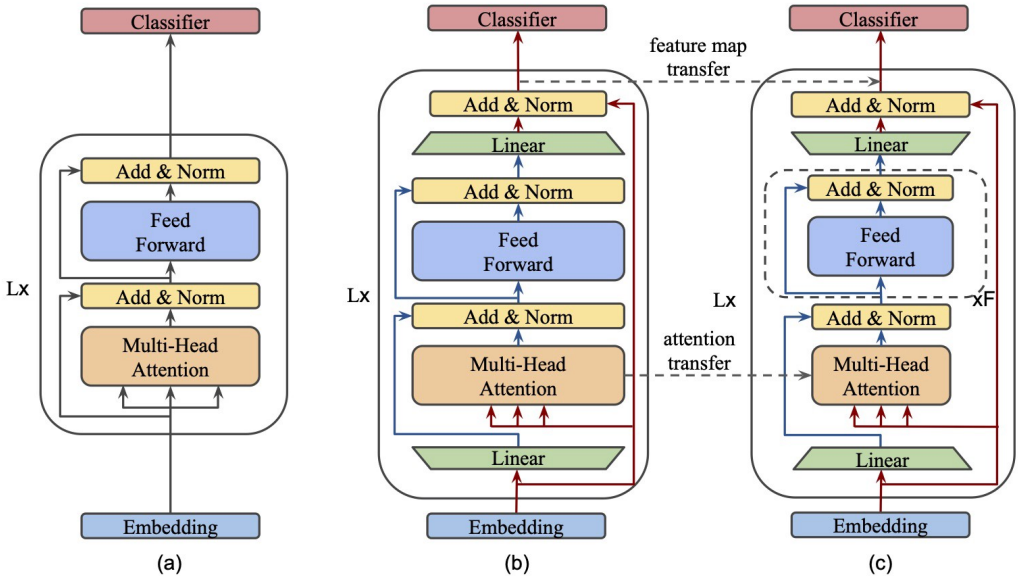

Figure 6. MobileBERT architecture. (**a**) BERT; (**b**) MobileBERT teacher; (**c**) MobileBERT student [64].

DistilBERT is a transformer model that is smaller and faster than BERT. DistilBERT is trained on a corpus previously in self-supervised mode with the base BERT model [65]. DistilBERT was pretrained on three targets: masked language modelling (MLM), loss of cosine embedding and distillation loss.

This model is one of the most efficient; in addition to needing few parameters for its training, the space required to store the model makes it ideal for applications on mobile devices.

For example, Saha et al. [66], collected data from text posted on social networks and classified how children are harassed by various comments online. The aim of this work was to show society the risks that children face with the use of communication applications. In [67], the authors used this model to detect comments that denote online aggression and conclude that the use of information from multiple depths increases the model's performance. In addition, in [68], the authors presented TopicBERT to optimise the computational cost of fine tuning for document classification, which was achieved by complementary learning of thematic and linguistic models in a unified framework.

6. Frameworks for Mobile Devices

Current tools on the market that allow for the implementation of machine learning models on mobile, which, as we have seen, allow the functionalities of applications to be expanded to other areas. In addition to the tools already mentioned, other tools provide functionalities to generate machine learning models for mobile devices, as shown in Table 5.

- TensorFlow Lite: Is a computational intelligence platform for local inference designed primarily for low-resource computing hardware such as mobile devices and embedded and edge systems. It enables on-device artificial intelligence by supporting programmers in running their models on relevant hardware and IoT devices [69]. This tool provides various methods of optimisation, compression and conversion of an ML model into a tflite format. This platform ensures data security through local device training without the need for an Internet connection [70].
- OpenCV: OpenCV is an open-source computer library developed in C and C++ on Linux, Windows and MacOS X with support for Python, Ruby Matlab and other languages [71]. It also supports mobile applications, which allows for the development of applications that require face recognition, object detection, image processing and manipulation, etc.
- The ML Kit by Google: Is a free mobile development SDK for Google's machine learning model in Android and iOS applications. It has features in its computer vision and natural language processing APIs. All ML Kit APIs run on the device, enabling real-time use cases. This also means that the functionality is available offline [72].
- Core ML: Is a machine learning development kit for Apple devices. It offers easy integration of machine learning models into applications. This library allows the user to transform models generated by other libraries using the ML Core utility, in addition to allowing users to preview the model directly from Xcode and download it using the ML Core Ready utility. It also allows the user to transform models from other types of libraries using Core ML Converters or downloaded ready-made Core ML models and preview the model easily or directly in Xcode. Furthermore, the kit allows the user to create computer vision, natural language, speech and audio models [73].
- Google Cloud AI: Tools use Google technologies to help developers solve common AI problems. Google AI continuously updates products and implemented algorithms in order to achieve the best inference results for developers. Google AI features include speech-to-text conversion, natural language processing and optical character recognition, among others [74].
- CAFFE2: Caffe2 provides an easy way to provide proof of concept and take advantage of the contributions of new models and algorithms provided by the scientific community. GPUs can be use in the cloud to train large volumes of data and scale trained models to mobile devices using Caffe2's cross-platform libraries [75].
- DialogFlow: Is a natural language understanding platform with which users can design a conversational user interface and embed it in a mobile or web application. It analyzes different file types of input such as text input or audio input, such as a voice recording, and can respond to users in different ways such as through text or an artificial voice [76].
- Microsoft Cognitive Services: Is an artificial intelligence (AI) service that bases its operation on sending data to a central server that is in charge of carrying out the training and returning the trained model to the source device. This service helps developers add cognitive intelligence to applications without prior AI knowledge or skills. Azure Cognitive Services enables developers to add functionality to their applications such as the ability to see, hear, speak and analyze [77].
- The Firebase ML Kit: Is a set of tools and services that focused on offering the developer powerful machine learning so that can be included in apps using an Android or iOS system. It has a set of APIs, also known as an application programming interfaces, that is cloud-enabled and allows the user to perform different actions, such as recognizing text, recognizing landmarks and image tagging [78].

Table 5. Tools for implementing machine learning applications on mobile devices.

Model	Features	Support	Functions
CAFFE2 [75]	Integration with mobile applications	C++, Python, Android, IOS	Training and testing on the device
OpenCV [79]	Integration with mobile applications; Facial recognition; Gesture recognition	C++, Java, Python, Android, IOS	Training and testing on the device
TensorFlow Lite [20,69]	Lightl Integration with mobile devices; Efficient	Android, IOS, RaspBerry Pi	Training and testing on the device
Google ML KIT [72]	Light; Integration with mobile devices; High speed of inference	Android, IOS	Training and testing on the device
DialogFlow [76,80]	Multichannel implementation; Advanced AI; State-based models; End-to-end administration	C, C#, Go, Java Node.js, Python	Inferences in the cloud
Microsoft Cognitive Services [77,81]	Computer vision; Speech recognition; Natural language understanding; Decision management	Python, Java, .NET, JS, GO, PHP	Inferences in the cloud
Core ML [73]	Creation of models; Pretrained or own models; No training allowed	IOS and converted models from other libraries	Model implementation; Pretrained on devices
Firebase ML Kit [78]	Text recognition; Image tagging; Object recognition and tracking; Language identification	Android, iOS	Inferences on the device or in the cloud
Google Cloud AI [74]	Speech to text; Natural language; Document AI	Java, Go, Python, Node.js	Inferences on the device or in the cloud
Pytorch Mobile [82]	Integration in mobile applications	iOS, Android and Linux	Training and testing on the device

7. Federated Learning

Federated learning is an ML environment in which different clients collaborate to learn a centralised model while keeping client data decentralised [83]. In this model, several users share data remotely to a server for centralized deep learning model training; this improves iteratively, meaning that the more data are shared by the clients, the better fit the model will have. The training and tuning process is simple; first, each client downloads the pretrained base model from the cloud, which is trained with the client's data (private or public), and a model summary is created with the new data. The new model configuration is encrypted and sent back to the server, performing the reverse process of encryption, and the model is integrated with the client data and the base model [84].

Federated learning allows mobile devices to collaboratively learn the shared prediction model so that all training data can be kept on the device, which helps train the machine learning algorithm while allowing each device to maintain its own private and local data. This technology provides pervasive machine learning solutions, as well as flexible and managed real-time data. Federated learning can be used for numerous tasks and contexts, including offline and online learning procedures for algorithms [85,86].

In order to ensure user privacy, when a model is deployed on a mobile device using federated Learning features, the device initially downloads the base model (A) from a remote server; when there is a data candidate for retraining, this model summarises the changes as a update to the base model (B), which is sent to refresh the core model via encrypted communication (C). This ensures that no user data, whether private or not, leave the device and that no individual updates are stored in the cloud [85] (see Figure 7).

There are three types of federated learning [87]:

- Vertical federated learning: Is applied in cases in which the datasets share the same sample space but have a different feature space, with training data vertically divided.
- Horizontal federated learning: Is proposed for architectures for which the participating customer datasets share the same type of characteristics but have different data samples, with the entire data set divided horizontally into data samples and assigned to two customers.
- Hybrid federated learning: Is applied when datasets from different customers have not only have different sample architectures but also share different feature architectures.

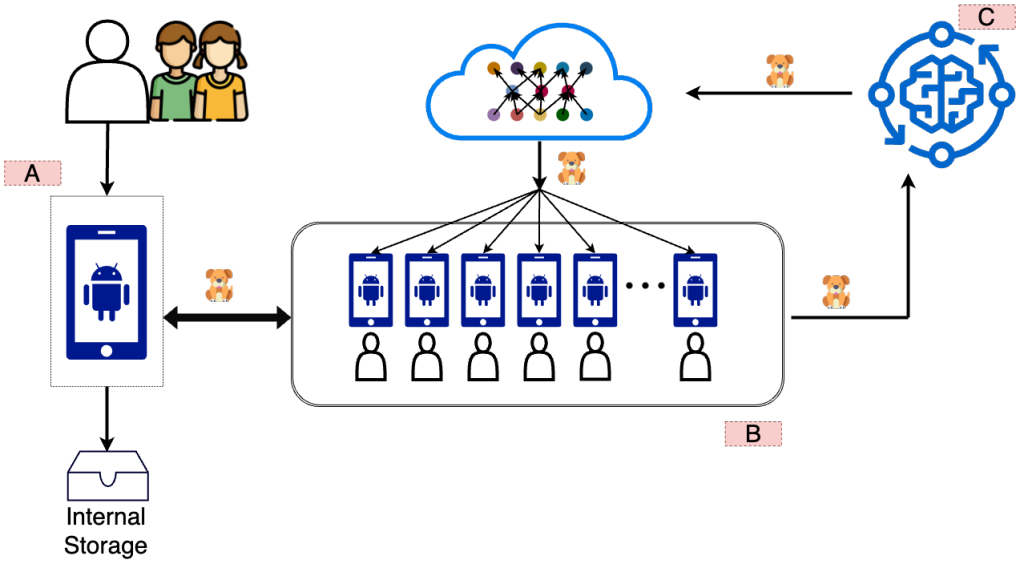

Figure 7. Functioning of federated learning. (**A**) the device downloads the base model, (**B**) summarises the changes as a update to the base model, (**C**) sent to refresh the core model via encrypted communication.

Federated learning has been used to improve the utility of various types of applications. Some of the most prominent applications of federated learning are Android's Gboard for predictive text and Google Assistant, in addition to natural language processing, autonomous vehicle, resource allocation, data science and health applications, among others [83].

Federated learning has the functionality to be incorporated into other industries, such as adding to customer financial records and adding to sound and image data. As more technology moves to mobile phones and other peripheral devices, federated learning provides a way to take advantage of streaming data [84].

8. Conclusions

In this paper, we analysed the literature on machine learning models that can be used to create more sophisticated and intelligent applications. We analysed the main architectures that a developer can use to implement machine learning models on mobile devices, taking into account the privacy and processing characteristics of the device. We also listed some of the most important frameworks that exist to implement AI models on devices with limited processing, comparing which of these can be used to make inferences either in the cloud or on the device. Since one of the main functionalities of AI applications is image classification and text analysis, some models and works were shown that adjust the parameters of the models so that they can be used in mobile devices, and a description of the concept of federated learning and its advantages with respect to developments in terms of privacy was provided. Finally, some challenges and problems currently encountered when implementing robust models on smartphones in terms of processing power, latency and memory were presented.

The technologies studied in this paper can be implemented on mobile devices to achieve certain tasks due to the increasing demand for new functionalities to satisfy the needs of users. With the implementation of artificial intelligence and machine learning algorithms, functionalities can be implemented that help users to carry out more specific activities in different areas, such as classifying images and videos, text analysis, prediction of possible words when writing, biometrics and mobile device software security, among others.In this work, it was possible to verify that for the implementation of machine learning algorithms in mobile devices, the size, memory, CPU, GPU and updates of the mobile device, as well as the security of sensitive information of a user, have to be taken into account.

Author Contributions: Conceptualization, S.P.A., A.L.S.O. and L.J.G.V.; methodology, S.P.A., A.L.S.O. and L.J.G.V.; validation, S.P.A., A.L.S.O. and L.J.G.V.; investigation, S.P.A., A.L.S.O. and L.J.G.V. All authors have read and agreed to the published version of the manuscript.

Funding: This work also was supported by the European Commission under the Horizon 2020 research and innovation programme, as part of the project HEROES (Grant Agreement no. 101021801). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or European Commission—EU. Neither the European Union nor the European Commission can be held responsible for them.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable; this study does not report any data.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sarwar, M.; Soomro, T.R. Impact of smartphone's on society. *Eur. J. Sci. Res.* **2013**, *98*, 216–226.
2. Statista. Share of Users Worldwide Accessing the Internet in 3rd Quarter 2022, by Device. Available online: https://www.statista.com/statistics/1289755/internet-access-by-device-worldwide/ (accessed on 10 April 2023).

3. Why On-Device Machine Learning? Available online: https://developers.google.com/learn/topics/on-device-ml/learn-more. (accessed on 15 January 2023).
4. Addressing the Challenges of On-Device Machine Learning. Available online: https://blog.developer.adobe.com/addressing-the-challenges-of-on-device-machine-learning-1f71ebcedd69 (accessed on 15 January 2023).
5. Addepto. What Are the Top 10 Challenges of Machine Learning? Available online: https://addepto.com/blog/what-are-the-top-10-challenges-of-machine-learning/ (accessed on 15 January 2023).
6. Linkedin. The Benefits and Challenges of Edge Machine Learning. Available online: https://www.linkedin.com/pulse/benefits-challenges-edge-machine-learning-wallaroolabs (accessed on 15 January 2023).
7. Ribeiro, M.; Grolinger, K.; Capretz, M.A. *MLaaS: Machine Learning as a Service*; IEEE: Piscataway, NJ, USA, 2015; pp. 896–902. [CrossRef]
8. Google Cloud, AI and Machine LEARNING Products. Available online: https://cloud.google.com/products/ai (accessed on 6 January 2023).
9. IBM Watson Machine Learning. Available online: https://www.ibm.com/cloud/watson-studio (accessed on 6 January 2023).
10. Welcome to Machine Learning Studio. Available online: https://studio.azureml.net/ (accessed on 6 January 2023).
11. Oracle Machine Learning. Available online: https://docs.oracle.com/en/database/oracle/machine-learning/oml4py/1/mlpug/machine-learning-classes-and-algorithms.html (accessed on 6 January 2023).
12. Machine Learning on AWS. Available online: https://aws.amazon.com/machine-learning (accessed on 6 January 2023).
13. Eshratifar, A.E.; Abrishami, M.S.; Pedram, M. JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services. *IEEE Trans. Mob. Comput.* **2021**, *20*, 565–576. [CrossRef]
14. Wang, X.; Xu, Y.; Xu, K.; Tagliasacchi, A.; Zhou, B.; Mahdavi-Amiri, A.; Zhang, H. PIE-NET: Parametric Inference of Point Cloud Edges. In Proceedings of the Advances in Neural Information Processing Systems (NeurIPS), Online, 6–12 December 2020; pp. 20167–20178.
15. Long, Y.; Chakraborty, I.; Srinivasan, G.; Roy, K. Complexity-Aware Adaptive Training and Inference for Edge-Cloud Distributed AI Systems. In Proceedings of the 2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS), Washington, DC, USA, 7–10 July 2021; pp. 573–583. [CrossRef]
16. Laskaridis, S.; Venieris, S.I.; Almeida, M.; Leontiadis, I.; Lane, N.D. SPINN: Synergistic Progressive Inferenceof Neural Networks over Device and Cloud. In Proceedings of the 26th Annual International Conference on Mobile Computing and Networking, MOBICOM '20, London, UK, 21–25 September 2020; Association for Computing Machinery (ACM): New York, NY, USA, 2020; pp. 1–15. [CrossRef]
17. Liu, Z.; Wu, Z.; Gan, C.; Zhu, L.; Han, S. DataMix: Efficient Privacy-Preserving Edge-Cloud Inference. In Proceedings of the Computer Vision—ECCV 2020: 16th European Conference, Glasgow, UK, 23–28 August 2020; Springer: Cham, Switzerland, 2020; Volume 12356, pp. 578–595. [CrossRef]
18. Ogden, S.S.; Kong, X.; Guo, T. PieSlicer: Dynamically Improving Response Time for Cloud-based CNN Inference. In Proceedings of the ACM/SPEC International Conference on Performance Engineering, Virtual Event, 19–23 April 2021; ACM: New York, NY, USA, 2021; pp. 249–256. [CrossRef]
19. Li, Y.; Han, Z.; Zhang, Q.; Li, Z.; Tan, H. Automating Cloud Deployment for Deep Learning Inference of Real-time Online Services. In Proceedings of the IEEE INFOCOM 2020—IEEE Conference on Computer Communications, Toronto, ON, Canada, 6–9 July 2020.
20. Reda, M.; Suwwan, R.; Alkafri, S.; Rashed, Y.; Shanableh, T. AgroAId: A Mobile App System for Visual Classification of Plant Species and Diseases Using Deep Learning and TensorFlow Lite. *Informatics* **2022**, *9*, 55. [CrossRef]
21. Mondal, S.; Modi, S.; Garg, S.; Das, D.; Mukherjee, S. *ICAN: Introspective Convolutional Attention Network for Semantic Text Classification*; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2020; pp. 158–161. [CrossRef]
22. Tan, M.; Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 6105–6114.
23. Vatsal, S.; Purre, N.; Moharana, S.; Ramena, G.; Mohanty, D. On-Device Information Extraction from Sms Using Hybrid Hierarchical Classification. In Proceedings of the 2020 IEEE 14th International Conference on Semantic Computing (ICSC), San Diego, CA, USA, 3–5 February 2020; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2020; pp. 178–181. [CrossRef]
24. Garg, S.; Harichandana, S.S.; Kumar, S. On-Device Document Classification using Multimodal Features. In Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD), Bangalore, India, 2–4 January 2021; Association for Computing Machinery: New York, NY, USA, 2020; pp. 203–207. [CrossRef]
25. Buiu, C.; Dănăilă, V.R.; Răduţă, C.N. MobileNetV2 ensemble for cervical precancerous lesions classification. *Processes* **2020**, *8*, 595. [CrossRef]
26. Ignatov, A.; Malivenko, G.; Timofte, R.; Tseng, Y.; Xu, Y.S.; Yu, P.H.; Chiang, C.M.; Kuo, H.K.; Chen, M.H.; Cheng, C.M.; et al. PyNet-V2 Mobile: Efficient On-Device Photo Processing With Neural Networks. In Proceedings of the 2022 26th International Conference on Pattern Recognition (ICPR), Montreal, QC, Canada, 21–25 August 2022; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2022; pp. 677–684. [CrossRef]

27. Sidhpura, J.; Shah, P.; Veerkhare, R.; Godbole, A. FedSpam: Privacy Preserving SMS Spam Prediction. In *Communications in Computer and Information Science, Proceedings of the ICONIP 2022: Neural Information Processing, New Delhi, India, 22 November 2022*; Springer: Singapore, 2023; Volume 1793, pp. 52–63
28. Nagula, P.K.; Alexakis, C. A new hybrid machine learning model for predicting the bitcoin (BTC-USD) price. *J. Behav. Exp. Financ.* **2022**, *36*, 100741. [CrossRef]
29. Wibowo, A.; Hartanto, C.A.; Wirawan, P.W. Android skin cancer detection and classification based on MobileNet v2 model. *Int. J. Adv. Intell. Inform.* **2020**, *6*, 135–148. [CrossRef]
30. GitHub. MobileNet. Available online: https://github.com/tensorflow/tfjs-models/tree/master/mobilenet (accessed on 8 January 2023).
31. Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; Chen, L.C. Mobilenetv2: Inverted residuals and linear bottlenecks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4510–4520.
32. Koonce, B. *Convolutional Neural Networks with Swift for Tensorflow*; Apress: Berkeley, CA, USA, 2021; pp. 109–123. [CrossRef]
33. Huang, J.; Mei, L.; Long, M.; Liu, Y.; Sun, W.; Li, X.; Shen, H.; Zhou, F.; Ruan, X.; Wang, D.; et al. Bm-net: Cnn-based mobilenet-v3 and bilinear structure for breast cancer detection in whole slide images. *Bioengineering* **2022**, *9*, 261. [CrossRef]
34. Michele, A.; Colin, V.; Santika, D.D. MobileNet Convolutional Neural Networks and Support Vector Machines for Palmprint Recognition. In *Procedia Computer Science, Proceedings of the 4th International Conference on Computer Science and Computational Intelligence (ICCSCI 2019): Enabling Collaboration to Escalate Impact of Research Results for Society, Yogyakarta, Indonesia, 12–13 September 2019*; Elsevier: Amsterdam, The Netherlands, 2021; pp. 110–117. [CrossRef]
35. Bi, C.; Wang, J.; Duan, Y.; Fu, B.; Kang, J.R.; Shi, Y. MobileNet based apple leaf diseases identification. *Mob. Netw. Appl.* **2020**, 1–9. [CrossRef]
36. Rajbongshi, A.; Sarker, T.; Ahamad, M.M.; Rahman, M.M. Rose Diseases Recognition using MobileNet. In Proceedings of the 2020 4th International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), Istanbul, Turkey, 22–24 October 2020; pp. 1–7. [CrossRef]
37. Zaki, S.Z.M.; Zulkifley, M.A.; Stofa, M.M.; Kamari, N.A.M.; Mohamed, N.A. Classification of tomato leaf diseases using MobileNet v2. *IAES Int. J. Artif. Intell.* **2020**, *9*, 290. [CrossRef]
38. Sae-Lim, W.; Wettayaprasit, W.; Aiyarak, P. Convolutional Neural Networks Using MobileNet for Skin Lesion Classification. In Proceedings of the 2019 16th International Joint Conference on Computer Science and Software Engineering (JCSSE), Chonburi, Thailand, 10–12 July 2019; pp. 242–247. [CrossRef]
39. Venkateswarlu, I.B.; Kakarla, J.; Prakash, S. Face mask detection using MobileNet and Global Pooling Block. In Proceedings of the 2020 IEEE 4th Conference on Information & Communication Technology (CICT), Chennai, India, 3–5 December 2020; pp. 1–5. [CrossRef]
40. Velasco, J.; Pascion, C.; Alberio, J.W.; Apuang, J.; Cruz, J.S.; Gomez, M.A.; Molina, B.J.; Tuala, L.; Thio-ac, A.; Jorda, R.J. A Smartphone-Based Skin Disease Classification Using MobileNet CNN. *Int. J. Adv. Trends Comput. Sci. Eng.* **2019**, *8*, 2632–2637. [CrossRef]
41. Souid, A.; Sakli, N.; Sakli, H. Classification and Predictions of Lung Diseases from Chest X-rays Using MobileNet V2. *Appl. Sci.* **2021**, *11*, 2751. [CrossRef]
42. Hartanto, C.A.; Wibowo, A. Development of Mobile Skin Cancer Detection using Faster R-CNN and MobileNet v2 Model. In Proceedings of the 2020 7th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE), Kota Semarang, Indonesia, 24–25 September 2020; pp. 58–63. [CrossRef]
43. Pan, H.; Pang, Z.; Wang, Y.; Wang, Y.; Chen, L. A New Image Recognition and Classification Method Combining Transfer Learning Algorithm and MobileNet Model for Welding Defects. *IEEE Access* **2020**, *8*, 119951–119960. [CrossRef]
44. Rahman, M.M.; Biswas, A.A.; Rajbongshi, A.; Majumder, A. Recognition of local birds of Bangladesh using MobileNet and Inception-v3. *Int. J. Adv. Comput. Sci. Appl.* **2020**, *11*, *309–316.* . [CrossRef]
45. Kadam, K.; Ahirrao, S.; Kotecha, K.; Sahu, S. Detection and Localization of Multiple Image Splicing Using MobileNet V1. *IEEE Access* **2021**, *9*, 162499–162519. [CrossRef]
46. Díaz-Gaxiola, E.; Morales-Casas, Z.E.; Castro-López, O.; Beltrán-Gutiérrez, G.; López, I.F.V.; Rendón, A.Y. Estudio comparativo de arquitecturas de CNNs en hojas de Pimiento Morrón infectadas con virus PHYVV o PEPGMV. *Res. Comput. Sci.* **2019**, *148*, 289–303. [CrossRef]
47. EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling. Available online: https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html (accessed on 10 January 2023).
48. EfficientNet Keras (and TensorFlow Keras). Available online: https://pypi.org/project/efficientnet/ (accessed on 6 January 2023).
49. Atila, Ü.; Uçar, M.; Akyol, K.; Uçar, E. Plant leaf disease classification using EfficientNet deep learning model. *Ecol. Inform.* **2021**, *61*, 101182. [CrossRef]
50. Yi, S.L.; Yang, X.L.; Wang, T.W.; She, F.R.; Xiong, X.; He, J.F. Diabetic Retinopathy Diagnosis Based on RA-EfficientNet. *Appl. Sci.* **2021**, *11*, 11035. [CrossRef]
51. Fudholi, D.H.; Rani, S.; Arifin, D.M.; Satyatama, M.R. Deep Learning-based Mobile Tourism Recommender System. *Sci. J. Inform.* **2021**, *8*, 111–118. [CrossRef]

52. GitHub. EfficientNet-Lite. Available online: https://github.com/tensorflow/tpu/blob/master/models/official/efficientnet/lite/ (accessed on 10 January 2023).
53. Blog, T. Higher Accuracy on Vision Models with EfficientNet-Lite. Available online: https://blog.tensorflow.org/2020/03/higher-accuracy-on-vision-models-with-efficientnet-lite.html (accessed on 15 January 2023).
54. Tan, M.; Le, Q. Efficientnetv2: Smaller models and faster training. In Proceedings of the 38th International Conference on Machine Learning, ICML, Online, 18–24 July 2021; pp. 10096–10106.
55. Howard, A.G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; Adam, H. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv* **2017**, arXiv:1704.04861.
56. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, Nevada, USA, 27–30 June 2016; pp. 770–778.
57. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
58. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the inception architecture for computer vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 2818–2826.
59. Zoph, B.; Vasudevan, V.; Shlens, J.; Le, Q.V. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 8697–8710.
60. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
61. Thompson, C.W.; Ross, K.M. Natural-Language Interface Generating System. U.S. Patent 4,688,195, 18 August 1987.
62. Sarker, S.; Wells, J.D. Understanding mobile handheld device use and adoption. *Commun. ACM* **2003**, *46*, 35–40. [CrossRef]
63. Jiao, X.; Yin, Y.; Shang, L.; Jiang, X.; Chen, X.; Li, L.; Wang, F.; Liu, Q. Tinybert: Distilling bert for natural language understanding. *arXiv* **2019**, arXiv:1909.10351.
64. Sun, Z.; Yu, H.; Song, X.; Liu, R.; Yang, Y.; Zhou, D. Mobilebert: A compact task-agnostic bert for resource-limited devices. *arXiv* **2020**, arXiv:2004.02984.
65. Sanh, V.; Debut, L.; Chaumond, J.; Wolf, T. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv* **2019**, arXiv:1910.01108.
66. Saha, U.; Mahmud, M.S.; Keya, M.; Lucky, E.A.E.; Khushbu, S.A.; Noori, S.R.H.; Syed, M.M. Exploring Public Attitude Towards Children by Leveraging Emoji to Track Out Sentiment Using Distil-BERT a Fine-Tuned Model. In Proceedings of the Third International Conference on Image Processing and Capsule Networks, Online, Bangkok, Thailand, 20–21 May 2022; pp. 332–346.
67. Palliser-Sans, R.; Rial-Farràs, A. HLE-UPC at SemEval-2021 Task 5: Multi-Depth DistilBERT for Toxic Spans Detection. *arXiv* **2021**, arXiv:2104.00639.
68. Chaudhary, Y.; Gupta, P.; Saxena, K.; Kulkarni, V.; Runkler, T.A.; Schütze, H. TopicBERT for Energy Efficient Document Classification. *arXiv* **2020**, arXiv:2010.16407.
69. For Mobile & Edge. Available online: https://www.tensorflow.org/lite/performance/model_optimization (accessed on 6 January 2023).
70. Rashidi, M. Application of TensorFlow Lite on Embedded Devices: A Hands-On Practice of TensorFlow Model Conversion to TensorFlow Lite Model and Its Deployment on Smartphone to Compare Model's Performance. Available online: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1698946&dswid=-2072 (accessed on 6 January 2023).
71. Brahmbhatt, S. *Practical OpenCV*; Apress: Berkeley, CA, USA, 2013; ISBN 978-1-4302-6080-6.
72. Google ML Kit. Available online: https://developers.google.com/ml-kit (accessed on 6 January 2023).
73. Machine Learning. Available online: https://developer.apple.com/machine-learning/ (accessed on 6 January 2023).
74. Google. Google Cloud AI. Available online: https://developer.apple.com/machine-learning/ (accessed on 6 January 2023).
75. CAFFE2. Caffe2: Anew Lightweight, Modular, and Scalable Deep Learning Framework. Available online: https://caffe2.ai/docs/caffe-migration.html (accessed on 6 January 2023).
76. Cloud, G. DialogFlow. Available online: https://cloud.google.com/dialogflow (accessed on 6 January 2023).
77. Del Sole, A. Introducing Microsoft Cognitive Services. In *Microsoft Computer Vision APIs Distilled: Getting Started with Cognitive Services*; Apress: Berkeley, CA, USA, 2018; pp. 1–4. [CrossRef]
78. Google. Firebase ML Kit. Available online: https://firebase.google.com/docs/ml-kit (accessed on 6 January 2023).
79. Bradski, G.; Kaehler, A. *Learning OpenCV: Computer Vision with the OpenCV Library*, 1st ed.; O'Reilly: Springfield, MO, USA, 2008; ISBN 978-0-5965-5404-0.
80. Reyes, R.; Garza, D.; Garrido, L.; la Cueva, V.D.; Ramirez, J. *Methodology for the Implementation of Virtual Assistants for Education Using Google Dialogflow*; Martínez-Villaseñor, L., Batyrshin, I., Marín-Hernández, A. Eds.; Springer International Publishing: Cham, Switzerland, 27 October 2019; Volume 11835, pp. 440–451. [CrossRef]
81. Masood, A.; Hashmi, A. *Cognitive Computing Recipes: Artificial Intelligence Solutions Using Microsoft Cognitive Services and TensorFlow*; Apress: Berkeley, CA, USA, 2019.
82. PyTorch. PyTorch Mobile. Available online: https://pytorch.org/mobile/home/ (accessed on 6 January 2023).
83. Banabilah, S.; Aloqaily, M.; Alsayed, E.; Malik, N.; Jararweh, Y. Federated learning review: Fundamentals, enabling technologies, and future applications. *Inf. Process. Manag.* **2022**, *59*, 103061. [CrossRef]
84. IBM. What Is Federated Learning? Available online: https://research.ibm.com/blog/what-is-federated-learning (accessed on 15 January 2023).

85. Research, G. Federated Learning: Collaborative Machine Learning without Centralized Training Data. Available online: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html (accessed on 15 January 2023).
86. Team, D.S. Aprendizaje Federado. Available online: https://datascience.eu/es/aprendizaje-automatico/aprendizaje-federado/ (accessed on 15 January 2023).
87. Zhu, H.; Zhang, H.; Jin, Y. From federated learning to federated neural architecture search: A survey. *Complex Intell. Syst.* **2021**, *7*, 639–657. [CrossRef]