ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/dheeraj-vaddepally/605

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 3 & Iss: 3

Research Article

Transfer Learning for Mobile-Based Computer Vision

Dheeraj Vaddepally*

Citation: Vaddepally D. Transfer Learning for Mobile-Based Computer Vision. *J Artif Intell Mach Learn & Data Sci 2025* 3(3), 2901-2906. DOI: doi.org/10.51219/JAIMLD/dheeraj-vaddepally/605

Received: 02 September, 2025; **Accepted:** 18 September, 2025; **Published:** 20 September, 2025

***Corresponding author:** Dheeraj Vaddepally, USA, E-mail ID: dheeraj.vaddepally@gmail.com

A B S T R A C T

Recent years have witnessed rapid growth in mobile-based computer vision applications. Deep learning advances have led to this, but the models have intensive computational and memory requirements that make their deployment on resource-constrained mobile devices challenging. Transfer learning offers a solution through the reuse of pre-trained models that reduce training time and data requirements. This paper explores how transfer learning has been applied for mobile-based computer vision, with particular emphasis on model development acceleration that is achieved by preserving performance. In particular, we discuss strategies to leverage a pre-trained model in overcoming some of the restrictions imposed by hardware on mobile. In addition to that, the paper highlights several best practices toward optimizing pre-trained models for application on mobile systems, including compressing models as well as how real-time performances can be optimized. We show through case studies and experimental results how transfer learning can balance efficiency with accuracy, making it an ideal approach for deploying computer vision models on mobile devices.

Keywords: Transfer learning, Mobile-based computer vision, Deep learning, Pre-trained models, Model Development acceleration, Resource-constrained devices, Hardware limitations, Model compression, Real-time performance optimization

1. Introduction

Mobile-based computer vision has become highly relevant in many applications, from facial recognition and augmented reality to real-time object detection and health monitoring. Advanced deep learning models are required for such applications, which are difficult to deploy on mobile devices because of their limited processing power, memory and battery capacity. Training deep learning models from scratch is computationally expensive and time-consuming, making it infeasible for mobile scenarios, especially when labeled data is scarce[1].

This technique of transfer learning is very efficient to face the above challenges since it allows developers to work with pre-trained models that were trained on very large datasets, which can easily be adapted for new tasks without much additional training. It reduces the amount of training time, resources needed and also the amount of labeled data required in building high performing computer vision models for mobile development[2].

This paper discusses the use of transfer learning in mobile-based computer vision in terms of how pre-trained models can be leveraged to overcome hardware constraints and speed up model development. We also discuss fine-tuning strategies that enable models to adapt to specific mobile tasks without losing efficiency. First, we'll share best practices about how to optimize pre-trained models for deployment on mobile through model compression, pruning and quantization techniques.

The paper is centered toward making a holistic contribution in the way transfer learning can empower mobile-based computer vision applications and also give insight into the practicalities of fine-tuning models for specific use cases.

2. Background

Transfer learning is the technique of reusing a pre-trained model initially trained on one task for a different but related task. Transfer learning has proven to be very effective in the field of computer vision because of large pre-trained models such as VGG, ResNet, Inception and MobileNet, which have been trained on very extensive datasets such as ImageNet[3]. It does incorporate hierarchical features very helpful for quite a range of visual tasks such as object detection, classification and object segmentation. So, the basic theory about transfer learning says that features or elements such as edges, shape and even texture, at which lower layers typically find objects in most big datasets to be learned effectively transferred to different new tasks do not need pretraining from scratch.

The importance of transfer learning in mobile-based computer vision lies mainly in the fact that direct training of models on a mobile platform is not possible because the computational resources are weaker on a smartphone or an embedded system compared to that of a desktop or a server environment. The necessity of training a deep learning model from scratch places heavy requirements on both the availability of high-performance hardware in the form of GPUs or TPUs and on substantial amounts of labeled data. Similarly, energy consumption cannot be light on the heads of mobile devices either. So, with transfer learning, one can avail themselves of the models learned from vast datasets and fine-tune those to a task-specific, relatively small dataset while simultaneously cutting down training time and resource usage.

Some models are designed for deployment on mobile and edge devices. These are MobileNet, EfficientNet and ShuffleNet. These models strike a balance between accuracy and efficiency, as they are light-weight architectures with fewer parameters and less computational requirements[2]. For instance, depth wise separable convolutions in MobileNet reduce the number of parameters and operations, making it very efficient in mobile deployment.

3. Related Work

These were some early computer vision tasks relying on hand-crafted features, including SIFT and HOG, for object detection and image classification. Then the discovery of deep learning, in particular the CNNs, turned the scene as now the model was allowed to learn the features from the raw pixel data automatically. AlexNet showed, in 2012, that it is possible to achieve high levels of accuracy with image classification by deep learning. These increased the level of accuracy further with subsequent models such as VGG and ResNet.

In computer vision, transfer learning made famous models like ResNet and Inception, that achieved good performances through fine-tuning for other purposes. Research evidence was that the models transferred from pre-trainings indeed saved much of the training time apart from increasing the rate of accuracy, even using fewer training data. We also found out that learned features can be transferred across different visual tasks and that deep layers in CNNs learn task-specific features while earlier layers capture general visual patterns[1].

In the recent past, there has been growing interest in deploying computer vision models on mobile and edge devices[2], a lightweight CNN designed for mobile applications. MobileNet reduced the computational burden by replacing traditional convolutions with depth wise separable convolutions. EfficientNet introduced a compound scaling method to balance model size, computation and accuracy while further optimizing models for mobile deployment[3].

Relevant literature reports approaches of optimizing pre-trained models for execution on mobile platforms, including model pruning, quantization and knowledge distillation. Deep Compression compresses deep learning models by pruning and quantization to enable deployment on resource-constrained devices[1]. Hinton introduced knowledge distillation as the ability of a network to transfer knowledge from a large, pre-trained model to a smaller one, which is efficient yet still performing[1].

Transfer learning, mobile-friendly model architectures and model optimization techniques form the backbone of modern computer vision in the mobile-based architecture. This paper expands further on these works by discussing strategy approaches for utilization of pre-trained models, tuning them to become mobile-specific in tasks and optimization techniques aimed at achieving the real-time response on mobiles. We position our work into the larger umbrella of mobile computer vision through review of related works and point to key contributions toward advancing the art in this domain.

4. Challenges in Mobile Based Computer Vision

One of the major problems with mobile-based computer vision is that it has extremely low computational power compared to the servers or desktops well-equipped with high-powered GPUs. Deep learning models can't run at their optimal complexity on these low-power processors of mobile devices and this reduces the overall efficiency of performance at the cost of accuracy.

The other limitation is that mobile devices are equipped with very few memory and storage options. Very big deep learning models can quickly overwhelm the available RAM and storage in a smartphone or a tablet. Another common technique in reducing the model size is known as model pruning - removal of unnecessary parameters. Another approach used is reducing the precision to the lower-bit format[4].

Another major problem is energy consumption due to the constant running of deep learning models draining the battery of a mobile device. Real-time applications, like augmented reality or video analytics, aggravate the problem since they have to make an inference constantly. Optimized models for mobile and energy-efficient inference frameworks are required to cut down on the power consumption but maintain acceptable performance.

The third challenge involves the real-time performance requirements of the applications. For applications such as augmented reality, autonomous navigation and real-time video processing, there is the need for very fast, low-latency inference for a good user experience. Models are mostly optimized for speed data processing and are accurate in most cases; however, such is achieved in several cases with the optimization of the input image resolution or with the use of hardware acceleration.

Mobile-based computer vision complicity further happens due to network dependency. Even though offloading computation on to cloud servers might help bypass the aspect of the limitation posed by mobile hardware, it has its dependency with network qualities, which causes lag or poor performance in those places that have poor connectivity. Further, uploading data to the cloud also brings up issues of privacy, especially with sensitive visual data. On-device processing counteracts these disadvantages but needs optimized lightweight models that can operate locally on a device.

Besides, with applications based on mobility, especially in those involving vision data, comes increased concern over the privacy of information. The prospect of breaches as well as breaches of privacy by sending user information to cloud servers for processing augurs ill for this aspect of applications. All inference is therefore done on a device without moving data to the external servers so as to preserve privacy and safety.

It is highly complex when developing and deploying computer vision models with huge diversity in the case of mobile hardware. In reality, they vary greatly when looking at their processing power, memory or their respective operating systems. Thus, carefulness in all configurations for such a scenario means testing it in hardware-specific optimizations so that it runs efficiently across a large variation of hardware[5].

The greatest challenge for generalization is across real-world environments. For example, taking pictures using a mobile phone will involve handling the different lighting conditions, the various angles or orientations and background, implying that computer vision models have to generalize on those accounts. The model which a specific dataset produces is actually not that robust when used in practice within various contexts. Thus, as such, the models have to be fine-tuned by specific data from specific domains for further strength in terms of robustness[5].

Mobile-based computer vision is a challenge of balancing pursuit of accuracy with efficiency. Deep, complex models have greater accuracy but tend to come at great costs in terms of resource utilization and will not run on hardware. Thus, lightweight architectures like MobileNet and EfficientNet-Lite find the optimal balance between competing demands that provide just enough accuracy with much-reduced usage of resources (Figure 1). The above problem requires a method that combines good model design along with optimization methods and effective learning transfer[4].

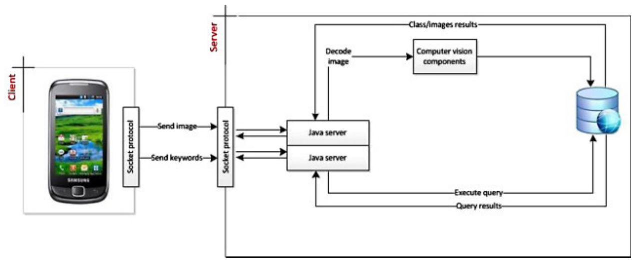

Figure 1: Mobile Based Computer Vision Architecture.

5. Leveraging Pre-Trained Models to Reduce Training Time

Most probably, the approach with the strongest positive impact in computer vision for mobile-based applications is the use of pre-trained models to obtain extremely fast training speed. Pretrained models are, in fact, strong foundations for learning new tasks- deep learning models previously trained with large, varied datasets such as ImageNet- and they greatly reduce the process required to train such a model compared to training them from scratch and resource consumption time.

Pre-trained models have a most significant advantage: they learn useful feature representations from an enormous amount of data. The early layers of deep neural networks are particularly prone to learning general features in computer vision; they learn edges, textures and shapes that generalize low-level features for many visual tasks and thus highly transferable[6]. This will greatly reduce the computational cost of training such features from scratch and saves much training time (Figure 2).

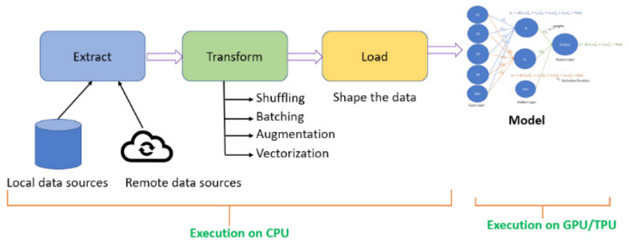

Figure 2: Reducing Training Time.

Fine-tuning is performed only on the last layers of a pre-trained model in mobile-based computer vision since the final layers of the model are specific to the task. This process is referred to as fine-tuning where only the last layers are trained again to accept the target task, such as object detection or facial recognition in the image classification of a mobile application[6]. The remaining network forms a generalized feature detector, which is not trainable during the course; it speeds up training and requires fewer data.

Pre-trained models are highly advantageous to use in mobile settings as the time and computational resources available are limited. Models for mobile and edge devices are those designed for use with MobileNet, EfficientNet and NASNet. Typically, the weights for such architectures come pre-trained. These are light architectures that require fewer parameters and less computation, but they also enjoy knowledge learned from general-purpose datasets in large-scale training.

Apart from the computational efficiency, there are practical advantages related to the reduced training time. For mobile application developers, quicker iterations and experimenting with faster cycles of training represent a matter of significant practical advantage[7]. Agility is essential in real-world applications where models might need to be quickly deployed, updated or adapted to new environments. This allows developers to speed up their development process and focus on fine-tuning their models for specific tasks rather than wasting much time training from scratch.

Another strength in the adoption of pre-trained models is smaller data sizes with which to train models. Once such a wealth of learned information exists in these models, one may fine-tune them easily using much smaller domain-specific sets, especially valuable for mobile applications in which datasets with many items can be pricey and time-consuming to collect as they require hand-labeled large data sets to get adequate performance.

5. Fine Tuning Strategies

Fine-tuning is one of the transfer learning processes most important in the case of computer vision for mobile-based applications. It simply takes a pre-trained model and adapts it to a new task by training it on a target dataset. Unlike training from scratch, which requires training all the layers of a network on a new task, fine-tuning targets modifying particular layers of a pre-trained model so that the performance can be preserved despite hardware constraints in the mobile device. Here are the essential strategies in fine-tuning mobile-based computer vision[7].

A. Freezing early layers

The most popular method of fine-tuning is freezing the early layers of the pre-trained model. Early layers usually learn low-level features, such as edges, corners and textures, which are commonly applicable to any task and dataset. Freezing those layers prevents retraining them, thus it saves computation and accelerates the fine-tuning process. The domain-specific features corresponding to the target task are fine-tuned only in later layers (Figure 4).

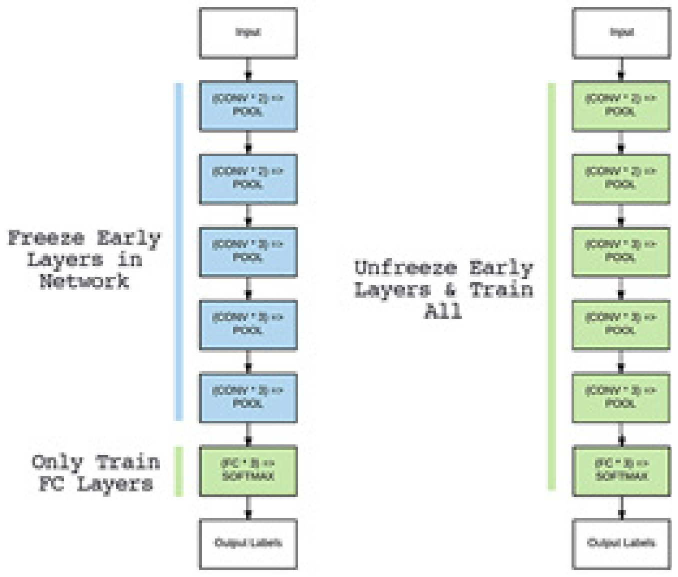

Figure 3: Freezing Early Layer.

Example: For transfer learning in facial recognition using MobileNet, the earlier layers are already quite effective at finding structures in the face. Fine-tuning is just retraining deeper layers in the hope of capturing specific properties of the faces in the new dataset.

B. Layer-wise learning rate adaptation

The learning rates can differ layer by layer inside a pre-trained model while it is undergoing fine-tuning. Lower layers have to be trained at a lower learning rate because they contain more general features while deeper layers, being task-specific, need a higher learning rate. This avoids the phenomena of overfitting inside lower layers and allows greater updates for better adaptation for the specific mobile task.

The first layers in the pre-trained ResNet model, which will be fine-tuned toward object detection, will have an update rate on a smaller level; however, some of the deeper layers that are utilizing classifications can be learned much more intensely, with hopes that high-performing success of the task will result.

C. Adding custom layers

To incorporate those in a suitable computer vision problem unique to a mobile-specific solution, new or customized layers at or past the last pretrained layer of these nets should generally be necessary before use in some optimal sense toward this end; some layers appended specifically include and would likely necessitate full or near full and well-conditioned classifications appropriate.

- **Example:** Consider an application which identifies various breeds of dogs; a pre-trained model like MobileNet can add a new dense layer followed by a softmax layer to classify breeds, with the original layers remaining responsible for feature extraction[8].

D. Partial fine-tuning

Since there are many computational and memory restrictions in domains with scarce resources, such as a mobile device, the model may not be fine-tuned all at once. Perhaps a more plausible direction is to perform partial fine-tuning. Here, all of the later layer's update, while preceding layers remain static. Partial fine-tuning reaches an excellent compromise between decreased requirements on computation time and performance boosts pertinent to tasks.

- **Example:** For mobile-based object detection using EfficientNet, instead of fine-tuning all layers, only the last few layers are retrained to adapt the model to the mobile dataset, which reduces memory consumption and training time[8].

E. Hyperparameter tuning for mobile tasks

Fine-tuning deals with hyperparameters, where optimization of the model is achieved toward mobile use. Some of these hyperparameters are batch size, learning rate and epochs in number. Due to memory, a small number of batches are required by a mobile device whereas a smaller number of learning may be needed since overfitting can be highly probable while learning (Figure 4). Determining the correct epoch number is quite important because small sizes of data get overfitted very soon.

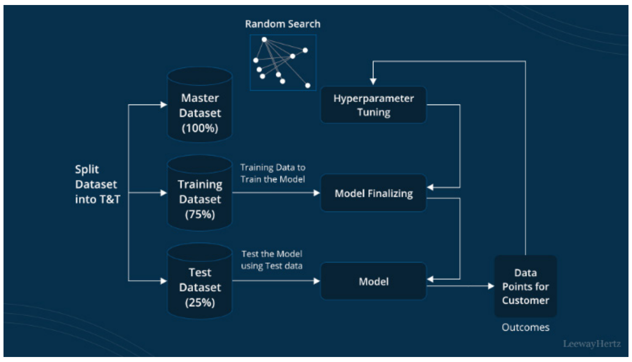

Figure 4: Hyperparameter tuning.

For instance, for plant disease detection in a mobile application, the batch size would be set equal to the available memory and fine-tuning would be performed with a reduced learning rate[8].

F. Regularization techniques

The fine-tuning model can avoid overfitting through dropout, L2 regularization and data augmentation. These improve the generalization ability of the model on the target dataset and result in robust performance on mobile devices. Dropout randomly "drops" some neurons during training, thereby making the model more generalized. Data augmentation such as random rotations, flips and zooms makes the model more robust to variations in input data[8].

For example, when fine-tuning the model, data augmentation applied may make it possible for the mobile application, detecting road signs, to be able to read the signs, irrespective of their angle or light conditions.

G. Transfer learning with pruned models

One advanced form of fine-tuning for mobile application is pretraining where a preexisting model with essential weights already eliminated to minimize size is considered the starting point; the pruning continues with subsequent fine-tuning on these pruned versions to reclaim most of their performances while not necessarily increasing to much larger size compared to mobile, which deems them unsuitable for such systems (Figure 5).

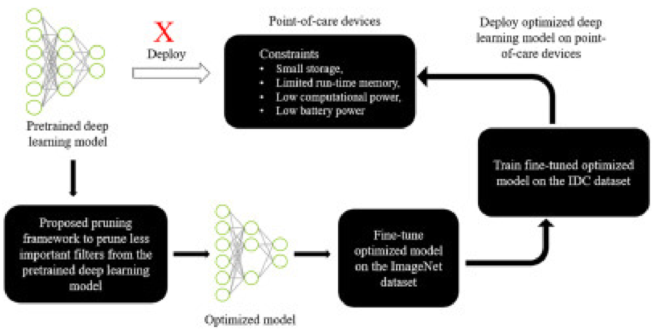
Pretrained deep learning model
X
Deploy
Point-of-care devices
Constraints
• Small storage,
• Limited run-time memory,
• Low computational power,
• Low battery power
Deploy optimized deep learning model on point-of-care devices
Train fine-tuned optimized model on the IDC dataset
Proposed pruning framework to prune less important filters from the pretrained deep learning model
Optimized model
Fine-tune optimized model on the ImageNet dataset

Figure 5: A transfer learning with structured filter pruning approach.

- **Example:** A version of MobileNet pruned could be fine-tuned on a task of image classification, finding that balance between high performance and computationally efficient model for mobile usage.

7. Best Practices for Transfer Learning in Mobile Vision Applications

Transfer learning for computer vision in mobile-based is the idea of using the pre-trained model to avoid spending time and resources in computing for deep learning models on a mobile device. The learned features will be applied directly to the mobile application, as the applications are able to utilize the knowledge of models learned in large datasets, such as ImageNet, rather than beginning anew. Fine-tuning models for a given task like object detection or image classification results in efficient deployment to mobile platforms. Freezing of layers, the application of quantization and usage of lightweight architectures are among such strategies for fine-tuning, optimizing the performance of the models on minimal consumption of resources within mobile environments[9].

- Right selection of pre-trained model: There is a model architecture that would be optimized for mobile deployment. This includes MobileNet, EfficientNet and even SqueezeNet. All of these are lightweight models with good performance levels, which can be deployed quite efficiently on a mobile device without compromising much in terms of performance[9].
- Limit depth of fine-tuning: Fine-tune just the last several layers of your model to tailor your model on a new task but without creating unnecessary overhead computation. These depth limits tend to decrease both the time for training and resource utilization, something really important and necessary for devices which have reduced computational powers on a mobile.
- Use quantization: This is the process of applying model quantization. It reduces the size of the model and helps improve inference speed. For example, changing from 32-bit floating-point operations to 8-bit integers or INT8, can highly reduce memory requirements and provide good accuracy in efficiency, mainly with quantization-aware training[10].
- **Leverage data augmentation:** Introduce data augmentation as a mechanism that enhances the generalization capability and robustness; in other words, there will be a higher capability of handling light and angle variations and any resolution on devices by introducing data augmentation in the training phase.
- **Resource monitoring:** Always monitor for inference time, memory footprint and battery consumption while the models are fine-tuned and deployed. The best practices highlighted above require that there be an excellent balance when it comes to achieving precision and efficiency in user experience in the mobile platform.

Transfer learning model adapts appropriately toward the computer vision application due to the best practices of above mobile as this brings together the precision, the speed and the resource constraint[10].

8. Case Study

A. Problem statement

A retail company wants to design a mobile application-based object detection app that can detect products on the shelves in stores, helping customers search and locate products much more speedily, monitor stock levels and conclude purchases. The challenge here is to implement a model that runs efficiently within the processing power and memory supplied by mobile devices, while still providing high accuracy in detecting a very wide variety of products at various lighting conditions and angles.

B. Solution

The team applied the concept of transfer learning by accelerating the development of the model while using the features of pre-trained models without having to train from scratch. The procedure undertaken was the following:

- **Model selection:** It chose the deployment of pre-trained MobileNetV2 for this purpose as it is quite efficient and smaller in size and will be better to deploy on the mobile platform. Features relating to texture, shapes and patterns are drawn out during pretraining of ImageNet on MobileNetV2.
- **Fine-tuning:** Fine-tuned on a custom dataset of images of all the products available in the store. Fine-tuned only the last few layers and not all the layers since freezing the initial layers which captures general features, drastically reduces the training time and consumption of resources.
- **Data augmentation:** Because the retail environment had different lighting conditions, angles and occlusions, random rotations, flips and color adjustments were applied during fine-tuning to increase model robustness and ensure that it could generalize well to real-world scenarios.
- **Quantization:** Further optimization for a mobile use

included applying quantization to reduce the size of the model. Applying quantization allowed the app to run faster on mobile devices, using less memory, to decrease battery drains while still maintaining at least acceptable levels of accuracy.

- **Testing on mobile:** The fine-tuned model was tested on iOS and Android devices. Inference speed, accuracy and memory usage were monitored against the requirements to ensure smooth functioning in a real-time setting of the app.

C. Outcome

The transfer learning technique enabled the team to rapidly develop an extremely efficient model for object detection, which would run flawlessly on mobile devices. The app correctly identified products with more than 90% accuracy, even in such challenging lighting conditions and clutters in the shelves. Mobile-specific optimizations through quantization and selection of lightweight models helped ensure that the app had excellent working performance on resource-poor devices, including very low memory usage and very fast inference times. Lessons Learned:

- The network was optimized more for mobile and had lighter versions like MobileNetV2 on top. A combination of some techniques, namely freezing of some layers and some form of quantization, gives the best compromise between performance and efficiency.
- **Importance of Data Augmentation:** Such diversity in natural product images requires data augmentation because otherwise, that model would go into overfitting and lower generalization capacities.
- **Continuous testing:** Continuous monitoring of the model on actual mobile devices helped fine-tune it further for optimal real-time use.

8. Conclusion

Transfer learning has emerged as the most important technique for effective and efficient deployment of computer vision models to mobile devices. This method is based on the application of pre-trained models which encoded knowledge from large-scale datasets and has resulted in an automatic reduction in training time and computational resources that are required in developing the model. This is particularly critical for the applications of mobile-based vision since they are constrained by hardware-related resources, such as memory, processing power and battery life.

We'll see some strategies that are employed in fine-tuning optimization over mobile devices. Some of them include freezing early layers, layer-wise learning rate adaptation and quantization, to name a few. Further on, we had best practices to select a light architecture and apply data augmentation techniques along with resource usage monitoring so that models' fast and accurate performance could be ensured over the mobile devices.

According to the case study, transfer learning helps mobile apps reach a degree of accuracy about objects detected with high precision within images, all classified, including many more sophisticated jobs, without employing an intense requirement for computational ability. With that, optimization along with transfer learning would make real-time experiences work within the device while making use of computer vision in mobile for users' preferences which match modern demands from mobile while operating within limited hardware of a smartphone.

In summary, transfer learning stands out as very powerful tool when it comes to mobile-based computer vision. For developers, that means building and deploying intelligent and efficient applications at scales across a plethora of devices, environments, on top of the growing mobile hardware of today. From this perspective and beyond, how advanced machine learning like transfer learning further determines the immediate future of these mobile vision applications is an increasingly important question of the day.

9. References

1. Yosinski J, Clune J, Bengio Y, et al. How transferable are features in deep neural networks?. Advances in neural information processing systems, 2014;27.
2. Cai H, Lin J, Lin Y, et al. Enable deep learning on mobile devices: Methods, systems and applications. ACM Transactions on Design Automation of Electronic Systems (TODAES), 2022;27: 1-50.
3. Tan M, Le Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In International conference on machine learning, 2019: 6105-6114.
4. Ogundokun RO, Awotunde JB, Akande HB, et al. Deep transfer learning models for mobile-based ocular disorder identification on retinal images. Computers, materials and continua, 2024;80: 139-161.
5. Benhamida A, Várkonyi-Kóczy AR, Kozlovszky M. Traffic Signs Recognition in a mobile-based application using TensorFlow and Transfer Learning technics. In 2020 IEEE 15th International Conference of System of Systems Engineering (SoSE), 2020: 000537-000542.
6. Dou S, Wang L, Fan D, et al. Classification of Citrus huanglongbing degree based on cbam-mobilenetv2 and transfer learning. Sensors, 2023;23: 5587.
7. Amugongo LM, Kriebitz A, Boch A, et al. Mobile computer vision-based applications for food recognition and volume and calorific estimation: A systematic review. In Healthcare, 2022;11: 59.
8. Casanova C, Franco A, Lumini A, et al. SmartVisionApp: A framework for computer vision applications on mobile devices. Expert systems with applications, 2013;40: 5884-5894.
9. Ramcharan A, McCloskey P, Baranowski K, et al. A mobile-based deep learning model for cassava disease diagnosis. Frontiers in plant science, 2019;10: 272.
10. Kargarandehkordi A, Washington P. Computer Vision Estimation of Stress and Anxiety Using a Gamified Mobile-based Ecological Momentary Assessment and Deep Learning: Research Protocol. medRxiv, 2023.