University of Science and Technology of Hanoi

Introduction to Deep Learning

Dr. Nghiem Thi Phuong & Dr. Tran Giang Son

Mid-term Report

MobileNet - A Case Study

BI12-447 An Minh Tri
BI12-389 Nguyen Son
BI12-375 Nguyen Cong Quoc
BI12-150 Tran Ngoc Hai
BI12-159 Pham Trung Hieu
BI12-445 Tang Minh Tri

Academic Year 3 - Data Science
November 2023

I. Introduction

Convolutional Neural Networks (CNNs) stand as a fundamental backbone within the domain of computer vision for a while now. They've come a long way from conventional 'vanilla' CNNs to more advanced architectures like LeNet, AlexNet, VGG, GoogleNet, and ResNet, …, each tailored with specific strengths for their specific tasks.

The evolution of CNNs has been significantly influenced by hardware developments like Graphics Processing Units (GPUs) and Tensor Processing Units (TPUs). Consequently, modern CNNs have grown in complexity, depth, and parameter count, to improve their accuracy. Since then, some models now have billions of parameters, making experiments theoretically impressive but practically challenging. [3]

But for now, imagine that you have to run these models on devices such as the Arduino kit, which has a less powerful GPU than a regular laptop, or use them on mobile devices like the smartphone in your pocket…it’s basically impossible.

So the big need now is for models that are small, lightweight, fast, and still pretty accurate. And that’s where MobileNets steps in!

MobileNets are lightweight models designed for efficient use on mobile and embedded devices. They are small, low-latency, low-power and have moderate accuracy, making them a great choice for real-world use.

They're a promising solution for deploying powerful models in situations where resources are limited, offering a smart way to handle complex tasks efficiently. [4]

II. Dataset

1. Data Understanding

In this demo, we'll use the CIFAR-10 Dataset. [5] The CIFAR-10 dataset, short for Canadian Institute for Advanced Research, was created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. It comprises 60,000 32x32 color images categorized into 10 distinct classes, with 6,000 images per class: Airplane, Automobile, Bird, Cat, Deer, Dog, Frog, Horse, Ship, Truck

Dataset Structure:

- **Training and Test Sets:** The dataset is divided into 50,000 training images and 10,000 test images.
- **Batch Division:** The dataset is segmented into five training batches and one test batch, each containing 10,000 images.
- **Class Distribution:** The test batch includes precisely 1,000 randomly selected images from each class. In contrast, the training batches encompass the remaining images, ensuring 5,000 images per class distributed randomly within the batches.

Class Distinctions:

- **Mutually Exclusive:** Each image belongs exclusively to one class without overlap between categories, ensuring distinct categorization.
- **Class Specifications:** For instance, the "Automobile" class encompasses sedans and SUVs but excludes pickup trucks, while the "Truck" class includes only large trucks, differentiating it from "Automobile."

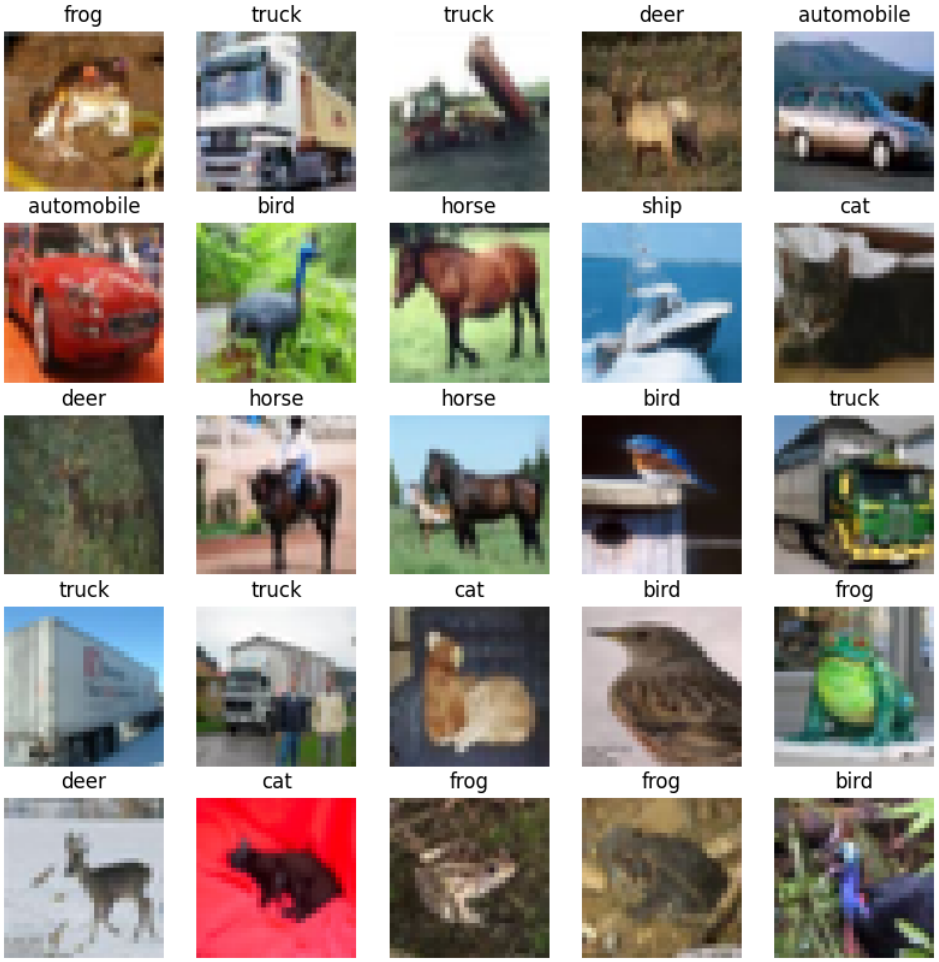
The CIFAR-10 dataset is widely used for image classification tasks and serves as a standard benchmark for evaluating machine learning models in computer vision, very suitable for training our lightweight models. [6]

2. Data Preparation

Dataset Description:

- *x_train:* Contains 40,000 color images sized 32x32 pixels each, with three color channels (RGB).
- *y_train:* Corresponds to 40,000 labels denoting classes for each image in x_train.
- *x_test:* Holds 10,000 color images, each sized 32x32 pixels, and containing three color channels (RGB).
- *y_test:* Includes 10,000 labels indicating classes for the images in x_test.

Data Visualization:



Dataset Split:

- **Training Set:** Consists of 80% of the data (40,000 images).
- **Validation Set:** Contains 20% of the data (10,000 images).

Data Preprocessing [6]:

- Image resizing to 32x32 pixels is not required for this dataset.
- Pixel values are normalized to a range between 0 and 1 by dividing them by 255.0.
- One-hot encoding applied to target labels:
    - Only binary values of 0 and 1 are used.
    - Each target label integer is transformed into a binary vector with a length equal to the maximum integer value found in the target labels.
    - The binary vector assigns 1 to the index corresponding to the class label and 0 to all other indices.

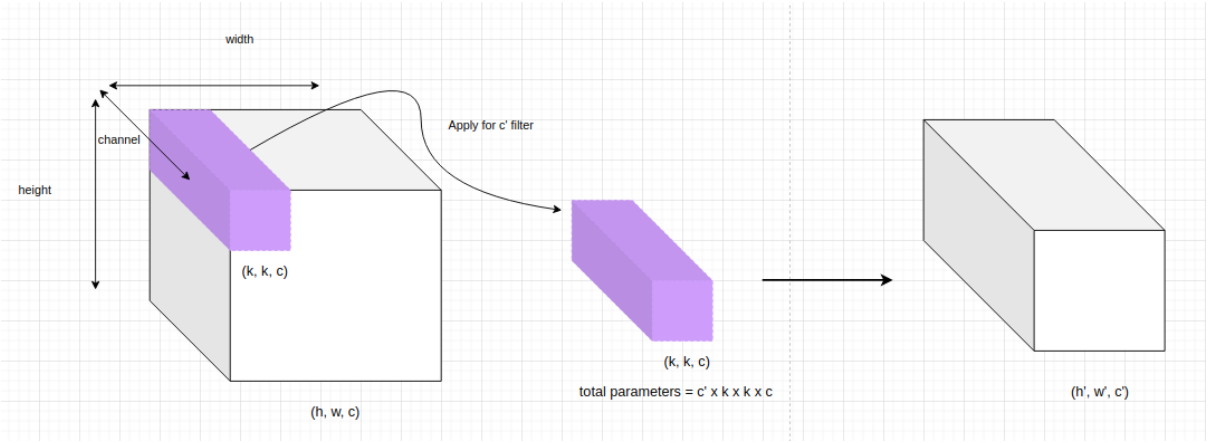
III. Model Architecture

To condense millions of parameters and reduce computational requirements while maintaining a relatively decent level of accuracy, MobileNets employed a mechanism known as Depthwise Separable Convolutions. But before that, let's take a quick look at how a typical Convolutional layer works.

1. Standard Convolutions

In short, a typical Convolutional layer operates by applying a set of filters to input data to the extract features. Each filter slides over the input, performing element-wise multiplications and summations to produce feature maps. The feature maps show important details like edges, textures, or patterns, which is needed to understand data in tasks like recognizing or processing images. The layer's parameters, including filter size, stride, and padding, influence the output dimensions and the information captured by the filters [1].

As we know, regular 2D convolutions are calculated across the entire depth (or channel) of a tensor. This means the number of parameters in the model significantly increases based on the depth of the preceding layer.



For example, with an input size of height (h) * width (w) * channels (c), a typical convolution operation requires k * k * c parameters to perform convolutions across the entire depth of the layers. Each filter creates an output matrix of size h' * w' * 1. Applying c' different filters results in an output of size h' * w' * c' (the output matrices from applying convolutions with each filter are concatenated along the depth). Therefore, the number of parameters needed for a regular convolution operation is c' * k * k * c. Imagine with today's high-resolution images, let's say 3x1024x1024, how extensive could the computational workload be? [12]

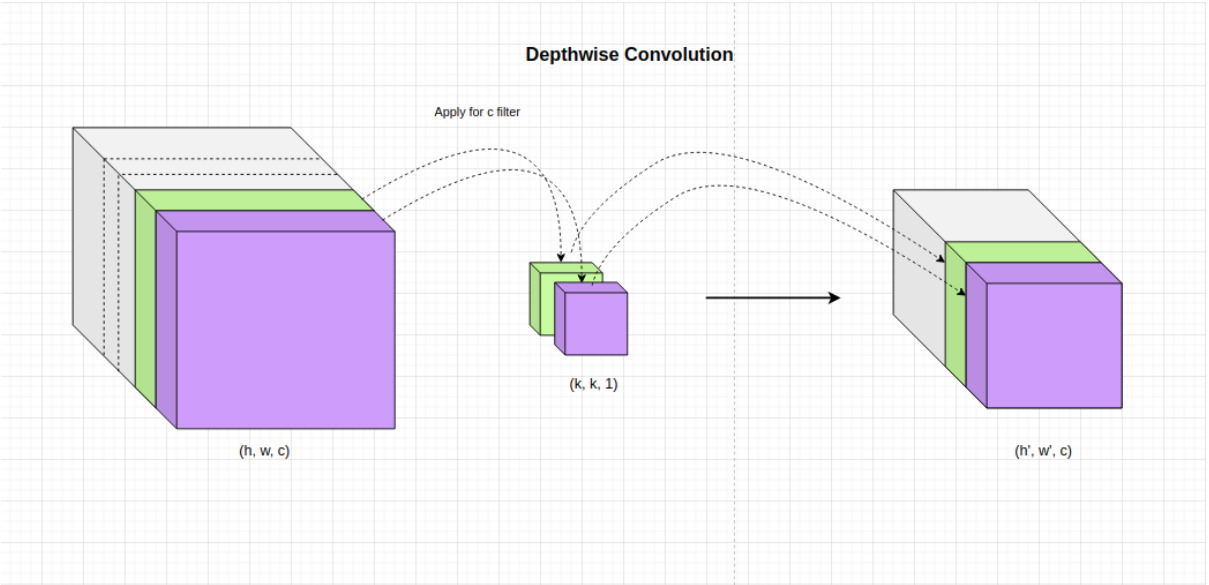
As the depth increases in CNNs, parameter counts increase, leading to memory-heavy models and slower computation. Models like AlexNet and VGGNet, reliant on standard 2D convolutions, notably amass large parameter counts. However, MobileNet models on the ImageNet Leaderboard possess fewer parameters, yet achieve better accuracy than AlexNet, thanks to MobileNet's pioneering use of

depth-wise separable convolutions. We'll delve into this architecture in the following section.

2. Depthwise Separable Convolutions

We recognize that depth is one of the main factors contributing to the increased number of parameters in a model. Depthwise separable convolution aims to eliminate depth dependency during convolution while still generating an output shape similar in size to that of a regular convolution. This process involves two sequential steps: [7]

2.1. Depthwise Convolution: The input tensor3D is divided into separate slices or matrices based on depth. Convolutions are performed on each slice independently, as illustrated below.

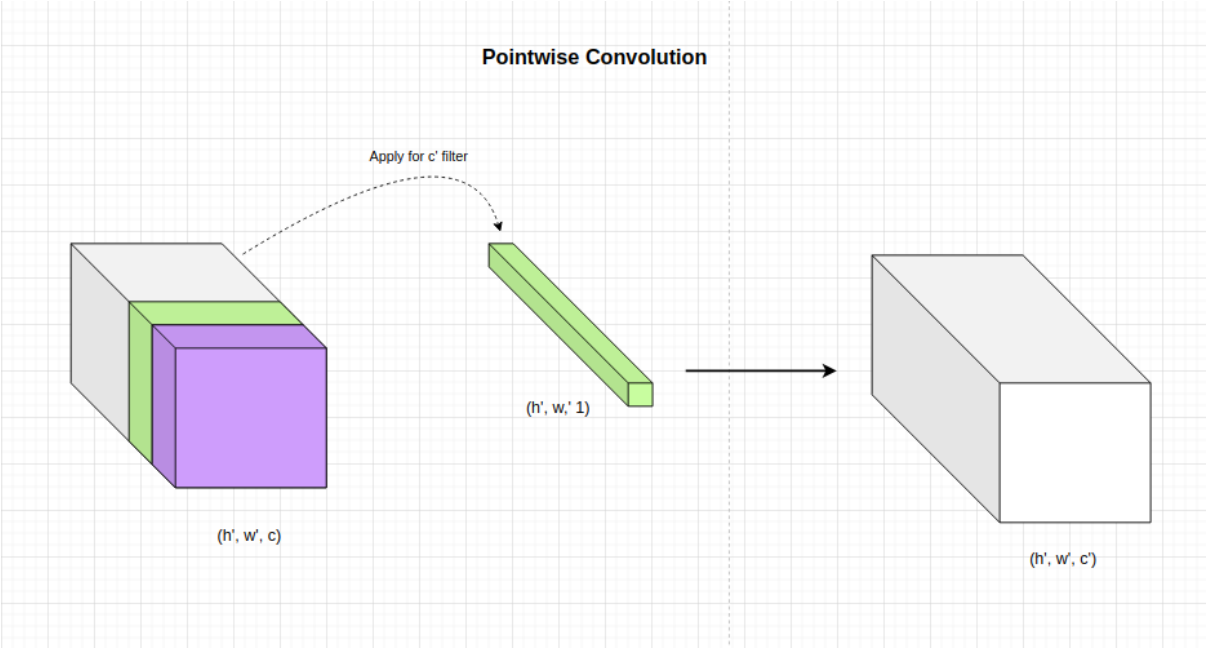


Each channel applies a different filter set, utilizing entirely unshared parameters. This has three primary effects on the model: [11]

- **Feature recognition:** Feature learning and recognition occur separately for each filter. Using dedicated filters for distinct features across channels enhances feature detection. For instance, in a three-channel RGB input, each channel employs a specialized filter.
- **Reduced computational load:** Traditional convolution requires k * k * c operations to generate one pixel in the output, whereas depthwise separable convolution requires only k * k operations.
- **Parameter reduction:** Depthwise convolution requires c * k * k parameters, significantly fewer (by a factor of c') compared to regular depthwise convolution.

The resulting convolutions are concatenated based on depth, producing a tensor3D block with dimensions h' * w' * c'.

2.2. Pointwise Convolution: This step modifies the depth from c to c'. It employs c' filters with a size of 1 * 1 * c, thus altering only the depth while keeping width and height constant.



The final output is a tensor with dimensions h' * w' * c'. The number of parameters required in this scenario is c' * c.

3. Comparing Depthwise Separable Convolution and Regular Convolution

The image below illustrates the difference between a standard CNN and the MobileNet architecture: [7]

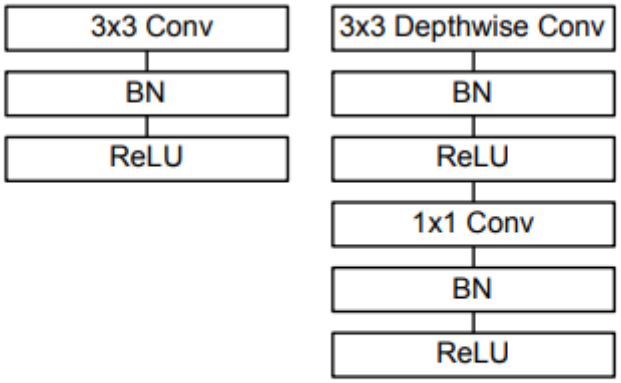


Figure 3. Left: Standard convolutional layer with batchnorm and ReLU. Right: Depthwise Separable convolutions with Depthwise and Pointwise layers followed by batchnorm and ReLU.

3.1. Parameter Count

As we know, to form a shape with dimensions h' × w' × c', the parameter count in regular convolution is c' × k × k × c. However, in depthwise separable convolution, it becomes k × k × c + c' × c. Usually, c' is larger than c because deeper layers have increased depth. Thus, the ratio:

$$\frac{c' \times k \times k \times c}{k \times k \times c + c' \times c} = \frac{c' \times k \times k}{k \times k + c'}$$

This ratio approaches k × k if c' is much greater than k × k. This significantly reduces the model size, explaining why MobileNet is considerably smaller compared to AlexNet.

3.2. Computational Operations

For creating an output shape with dimensions h' × w' × c', regular convolution requires (h' × w' × c') × (k × k × c) operations, where h' × w' × c' is the number of pixels needed, and k × k × c represents the multiplications to generate one pixel.

In depthwise separable convolution, the operations are sequentially performed as follows:

Depthwise Convolution: (h' × w' × c) × (k × k) multiplications.

Pointwise Convolution: (h' × w' × c) × (h' × w') multiplications.

The ratio of operations between regular and depthwise separable convolutions is

$$\frac{h' \times w' \times c' \times k \times k \times c}{h' \times w' \times c \times k \times k + h' \times w' \times c \times h' \times w'} = \frac{c' \times k \times k}{k \times k + h' \times w'}$$

This significant ratio shows that depthwise separable convolution requires much fewer computational operations compared to regular convolution, making it suitable for low-configured devices.

4. Model I/O

Input Layer: 32x32x3 (RGB Image)

Output Layer:

- **Classes:** 10 (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck)
- **Global Average Pooling 2D** (GAP):
    - The GAP layer condenses the spatial dimensions of the feature maps into a single vector by computing the average value of each channel.
    - This technique reduces the number of model parameters compared to traditional fully connected layers.
    - It enhances the model's ability to generalize by making it less sensitive to precise spatial locations of features, rendering it suitable for transfer learning.
- **Softmax Activation Function:**
    - Softmax is employed for multi-class classification tasks, converting the model's output logits into a probability distribution across the target classes.
    - It assigns probabilities to each class, allowing the model to predict the class with the highest probability.
    - This activation ensures that the sum of predicted probabilities equals 1, simplifying the interpretation of the model's output as class probabilities.

This architecture enables efficient feature extraction through global average pooling while facilitating accurate multi-class classification via the softmax activation function. [5]

IV. Evaluation

1. Evaluation Metrics

Test Loss:

- **Definition:** Test loss quantifies the average error or discrepancy between the model's predictions and the true labels within the test dataset.
- **Cross-Entropy Loss:** Utilizes cross-entropy as the chosen loss function, assessing the disparity between predicted class probabilities and actual class labels. This metric measures how well the model's probabilities align with the true class distributions. [10]

$$L_{\mathrm{CE}} = -\sum_{i=1}^{n} t_i \log(p_i), \text{ for n classes,}$$

where t_i is the truth label and p_i is the Softmax probability for the i^{th} class.

Test Accuracy:

- **Definition:** Test accuracy signifies the proportion of correctly predicted samples in the test dataset, illustrating the model's proficiency in accurate classification. [9]
- **Significance:** It showcases the model's ability to classify samples correctly and serves as a fundamental metric to evaluate classification performance.

$$\text{Accuracy} = \frac{\text{True Positives + True Negatives}}{\text{All Samples}}$$

These metrics, test loss and test accuracy, offer a comprehensive evaluation of the model's predictive performance and its capability to correctly classify samples while quantifying its predictive errors.

2. Evaluation

Upon training the MobileNet models, we obtained two model files: *wl_mobilenet_model.h5* for the model we trained from the beginning and *pr_mobilenet_model.h5* for the model we trained from the pretrained MobileNets model from the ImageNet dataset. Here's the evaluation of both models:

Our Trained-from-scratch Model Performance:

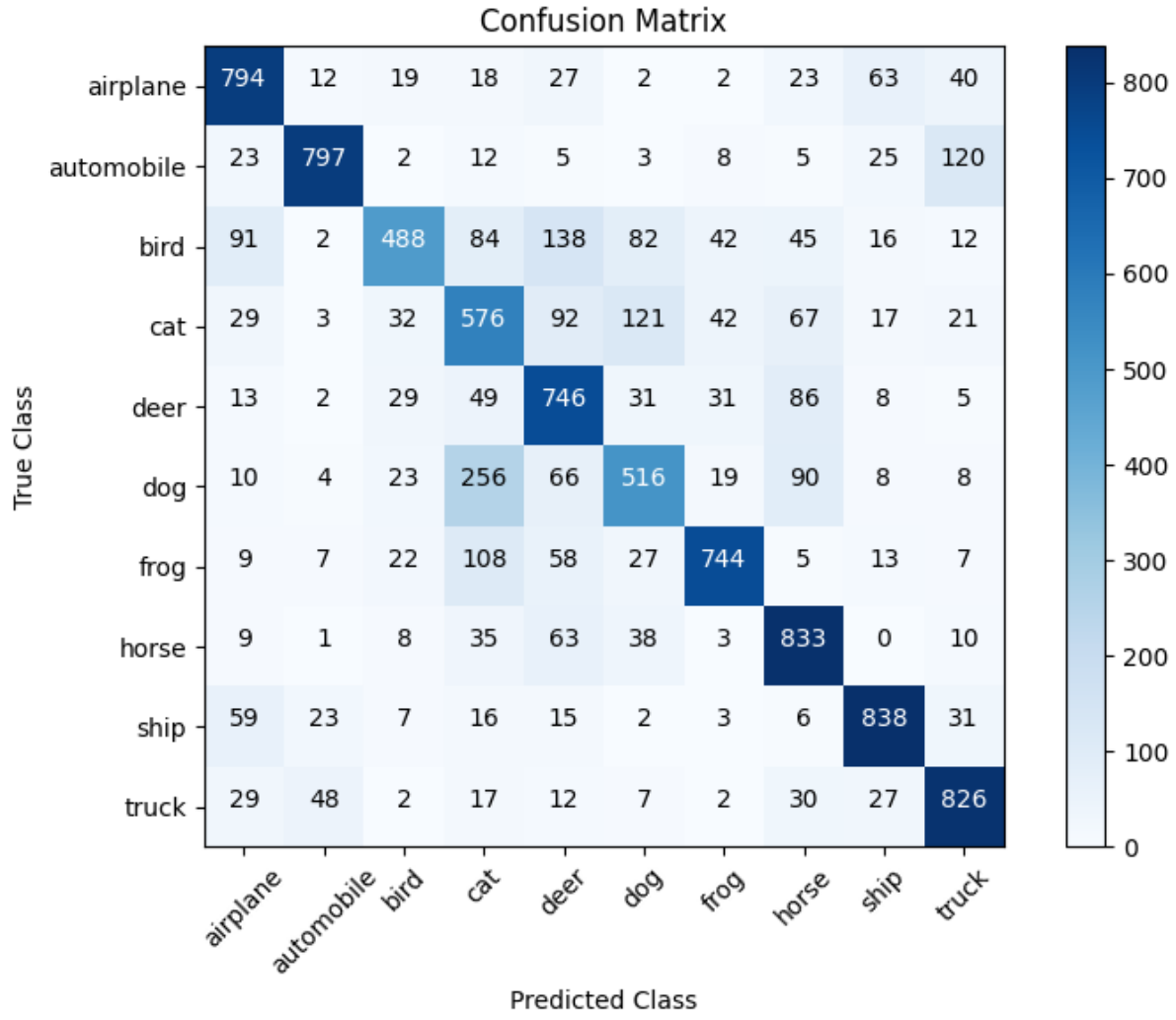
Test Loss: **0.9781020283699036**

Test Accuracy: **0.7157999873161316**

Our version of the MobileNet model achieved an accuracy of approximately 71.58% on the test dataset.

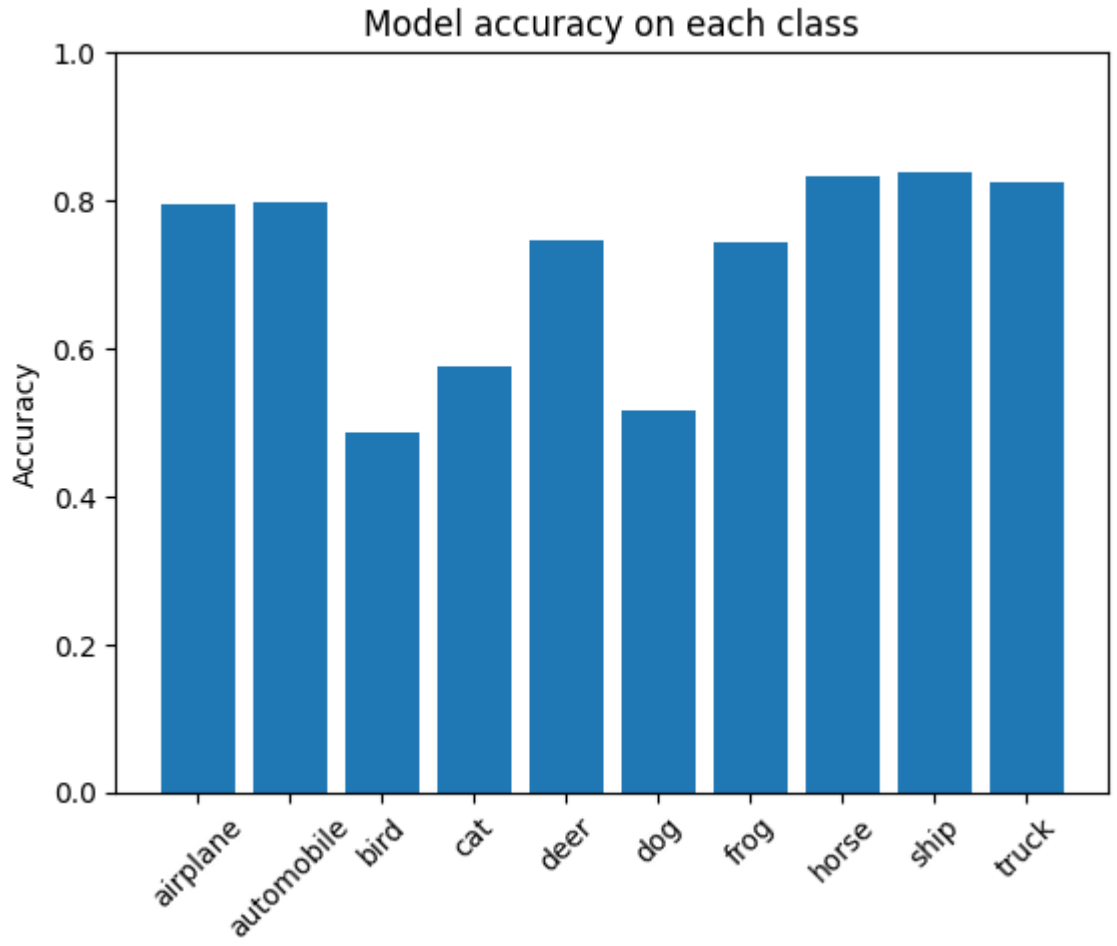
Confusion Matrix:

The confusion matrix below demonstrates the model's performance across different classes in the CIFAR-10 dataset. Each cell represents the count of instances where a class was predicted compared to the actual class.



Accuracy per Class:

Analyzing accuracy per class reveals variations in the model's performance across different categories. The bar chart demonstrates how accurately the model predicted each class within the dataset.



The ImageNet Dataset Pretrained model Performance:

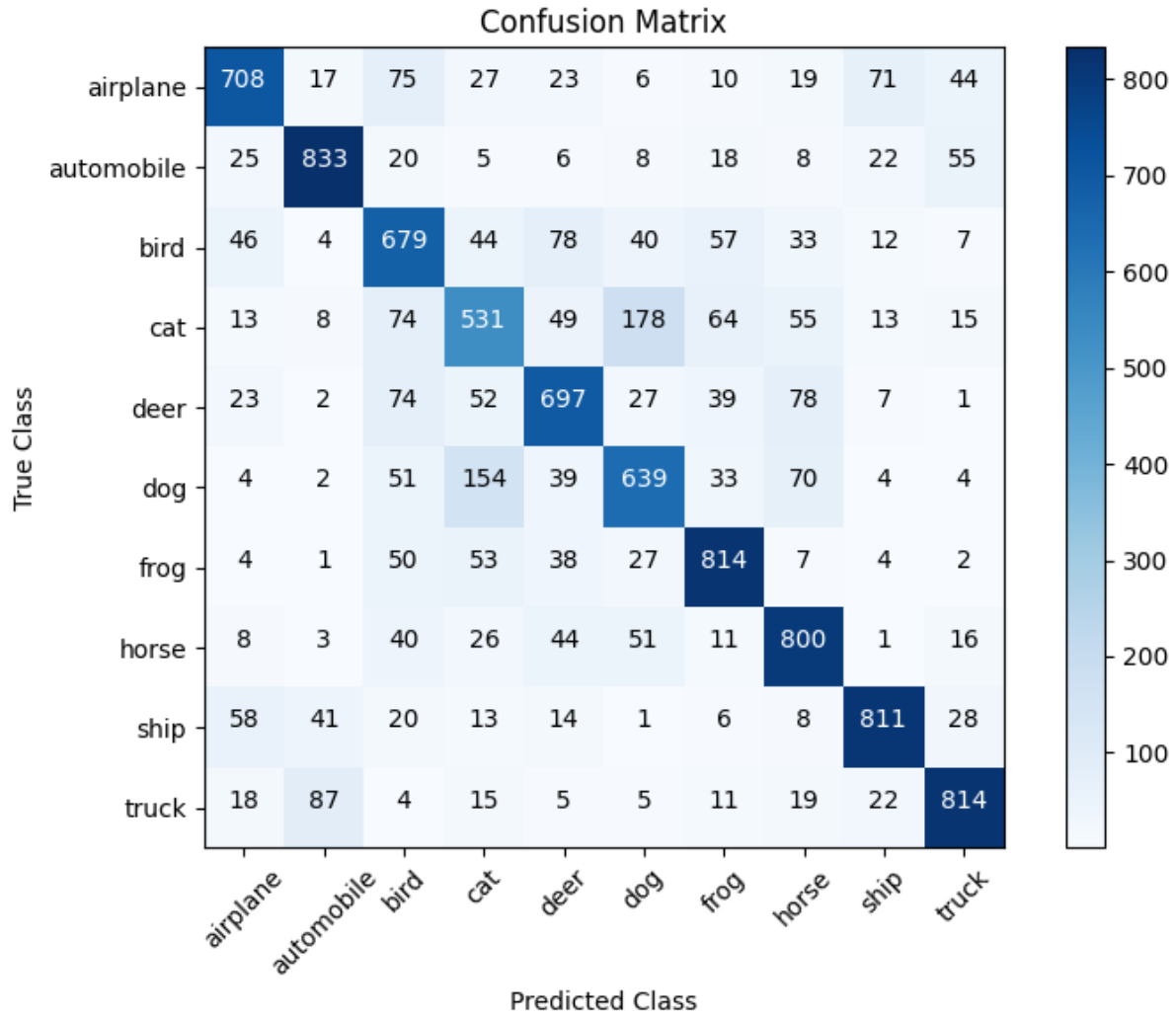
Test Loss: **0.8180649876594543**

Test Accuracy: **0.7325999736785889**

The pretrained MobileNet model exhibited an accuracy of about 73.26% on the test dataset.

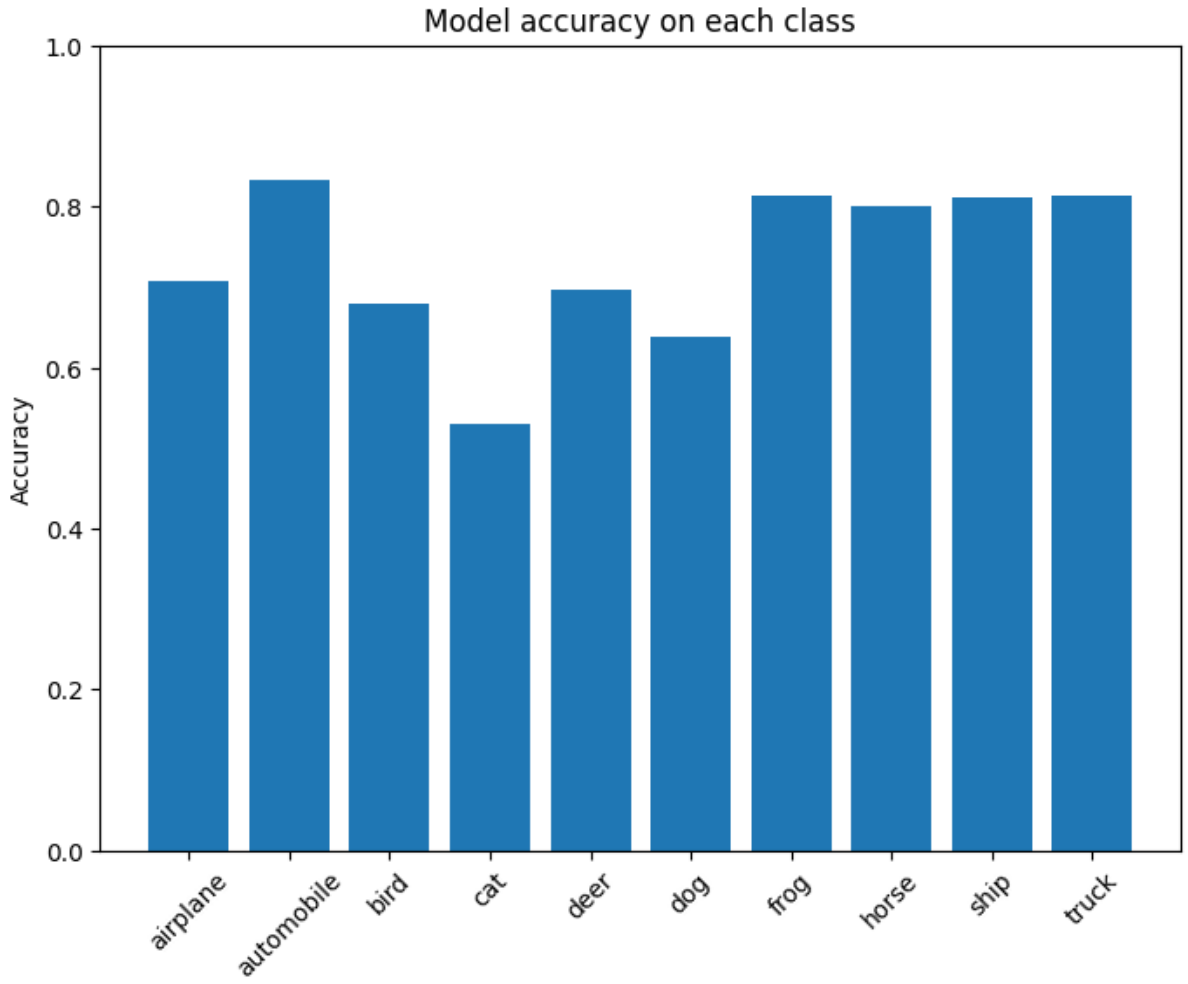
Confusion Matrix:

Similar to our model, the confusion matrix for the pretrained MobileNet model is depicted below, showcasing its performance across CIFAR-10 classes.



Accuracy per Class:

The bar chart illustrating accuracy per class demonstrates the pretrained model's performance across individual categories in the dataset.



These evaluations show that the ImageNet dataset pretrained MobileNet model performs slightly better than our version of the MobileNet model, achieving an accuracy of approximately 73.26% compared to 71.58%. The confusion matrices and accuracy per class visuals provide further insights into the models' strengths and weaknesses across different categories within the CIFAR-10 dataset

V. Demo

The following demonstration showcases image classification using the two MobileNet models that we've trained to predict the contents of an input image.

Loaded Image:

Here's a random image of an airplane: [8]

After the preprocessed and resized part, the original image turned out to be this:

Model Evaluation:

Two variations of MobileNet models were utilized for prediction:

Weight-less MobileNet Model

- **Predicted Class:** airplane
- **Confidence:** 99.94%
- **Other Predictions:** horse (0.05%), bird (0.01%), ship (0.00%), cat (0.00%), deer (0.00%), dog (0.00%), truck (0.00%), frog (0.00%), automobile (0.00%)

```
1/1 [==============================] - 0s 25ms/step
Predicted class: airplane, score: 99.939501%
Predicted class: horse, score: 0.052574%
Predicted class: bird, score: 0.007598%
Predicted class: ship, score: 0.000172%
Predicted class: cat, score: 0.000145%
Predicted class: deer, score: 0.000010%
Predicted class: dog, score: 0.000001%
Predicted class: truck, score: 0.000001%
Predicted class: frog, score: 0.000000%
Predicted class: automobile, score: 0.000000%
```

Pretrained MobileNet Model

- **Predicted Class:** airplane
- **Confidence:** 99.35%
- **Top Predictions:** deer (0.28%), bird (0.15%), truck (0.07%), horse (0.05%), ship (0.03%), automobile (0.03%), dog (0.03%), cat (0.02%), frog (0.00%)

```
1/1 [==============================] - 0s 36ms/step
Predicted class: airplane, score: 99.353856%
Predicted class: deer, score: 0.282742%
Predicted class: bird, score: 0.151999%
Predicted class: truck, score: 0.065409%
Predicted class: horse, score: 0.045953%
Predicted class: ship, score: 0.028773%
Predicted class: automobile, score: 0.027289%
Predicted class: dog, score: 0.025848%
Predicted class: cat, score: 0.015308%
Predicted class: frog, score: 0.002829%
```

The prediction results indicate that both models identified the image as an "airplane" with high confidence scores, demonstrating the effectiveness of the MobileNet architecture in classifying images. However, slight variations in confidence scores and alternative class predictions were observed between the weight-less and pretrained models.

The pretrained model displays slightly lower confidence in its prediction compared to the weight-less model but still correctly identifies the image as an "airplane." These predictions highlight the models' capability to recognize objects within images and showcase their respective levels of confidence in predictions.

VI. References

[1] Gurucharan, M.K. (2022) *Basic CNN Architecture: Explain 5 Layers of Convolutional Neural Network.* upGrad. Retrieved from: https://www.upgrad.com/blog/basic-cnn-architecture/

[2] AI & Insight. (2023) *Convolutional Neural Networks (CNNs) in Computer Vision.* Medium. Retrieved from: https://medium.com/@AIandInsights/convolutional-neural-networks-cnns-in-computer-vision-10573d0f5b00

[3] Srudeep, P.A. (2020). *An Overview on MobileNet: An Efficient Mobile Vision CNN*. Medium. Retrieved from: https://medium.com/@godeep48/an-overview-on-mobilenet-an-efficient-mobile-vision-cnn-f301141db94d

[4] *Understanding of MobileNet - EN*. (2023). 위키독스. Retrieved from: https://wikidocs.net/165429

[5] Krizhevsky, A., Vinod, N., and Geoffrey, H. (2023). *CIFAR-10 and CIFAR-100 datasets*. cs.toronto.edu. Retrieved from: https://www.cs.toronto.edu/~kriz/cifar.html

[6] Krizhevsky et al. (2022). *CIFAR-10 Dataset*. Paperswithcode. Retrieved from: https://paperswithcode.com/dataset/cifar-10

[7] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., & Adam, H. (2017). *MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications*. ArXiv. Retrieved from: https://arxiv.org/abs/1704.04861

[8] Walton, J. (2023, January 3). *The new aircraft, routes and airplane cabins taking off in 2023*. CNN. Retrieved from: https://edition.cnn.com/travel/article/aviation-lookahead-2023/index.html

[9] *4 things you need to know about AI: accuracy, precision, recall and F1 scores*. (2019). Lawtomated. Retrieved from: https://lawtomated.com/accuracy-precision-recall-and-f1-scores-for-lawyers/

[10] Kiprono Elijah Koech. (2020). *Cross-Entropy Loss Function.* Medium; Towards Data Science. Retrieved from: https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

[11] When Maths Meet Coding. (2020). *depthwise separable convolution | Complete tensor operations for MobileNet architecture* [YouTube Video]. *YouTube*. Retrieved from: https://www.youtube.com/watch?v=vfCvmenkbZA&t=256s

[12] Animated AI. (2023). *Groups, Depthwise, and Depthwise-Separable Convolution (Neural Networks)* [YouTube Video]. *YouTube*. Retrieved from: https://www.youtube.com/watch?v=vVaRhZXovbw