Convolutional neural networks for image classification on CIFAR-10 and CIFAR-100

Sanmar Simon
École Polytechnique Montréal, Canada
sanmar-yared.simon@polymtl.ca

Abstract

The aim of this project is to study the performance of five convolutional neural networks (CNNs) pre-trained on ImageNet for image classification on the CIFAR-10 and CIFAR-100 datasets. The CNNs used in this study are VGG16, EfficientNetV2_M, EfficientNet_B0, ResNet-50 and DenseNet-121. We refined these pre-trained models on the CIFAR-10 and CIFAR-100 datasets and evaluated their per- formance using various measures such as accuracy, precision, recall and F1 score.
Our results showed that all the pre-trained models achieved high accuracy on both datasets, with EfficientNetV2_M outperforming the others on both the CIFAR-10 and CIFAR-100 datasets. We also analyzed the computational resources required to train each model and found that EfficientNetV2_M required fewer resources for training, while VGG16 needed more.
Overall, this study highlights the importance of selecting an appropriate pre-trained model for a given image classification task, as well as the importance of taking computational effi- ciency into account when fitting these mo- dels.

1 Introduction

Image classification is a fundamental task in computer vision, and the development of deep learning techniques has revolutionized the field. Convolutional neural networks (CNNs) have established themselves as the state-of-the-art models f o r image classification, and numerous pre-trained CNN models are available for fine-tuning on specific data sets. In recent years, several studies have explored the performance of pre-trained CNNs in image classification tasks, notably on the CIFAR-10 and CI- FAR-100 datasets.

For example, a study by [He et al., 2016] examined the performance of several CNN architectures, including ResNet, on the CIFAR-10 and CIFAR- datasets.
100. Their results showed that ResNet outperformed other CNN models on both datasets, reaching peak accuracy. Similarly, a study by [Tan and Le, 2019] proposed a new family of CNN models called EfficientNet, which achieved peak accuracy on several image classification benchmarks, including CIFAR-10.

Other studies have explored the performance of pre-trained CNNs on more specialized image classification tasks. For example, a study by [Kumar et al., 2021] examined the performance of pre-trained CNN models for the classification of diabetic retinopathy, achieving high accuracy using finely-tuned models. Another study by [Liao et al., 2021] explored the per- formances of pre-trained CNNs for COVID-19 classification from chest X-ray images.

In the present study, we build on this previous work and examine the performance of five different pre-trained CNNs, including VGG16, Efficient- NetV2_M, EfficientNet_B0, ResNet-50 and DenseNet-121, on the CIFAR-10 and CIFAR-100 datasets. Our aim is to determine which pre-trained CNNs are most effective for image classification on these datasets, and to give an insight into the computational res- sources needed to train these models.

2 Theoretical approach

In this section, we'll look at the theory behind convolutional neural networks, the specific features of the models we've chosen, and the approach to fitting a pre-trained model.

2.1 Convolutional neural networks

The basic architecture of a convolutional neural network consists of the following three layers.

Convolutional layer
The convolutional layer is the core element of a CNN. It applies a set of learnable filters (also called kernels or weights) to the input data.
Each filter glides over the input data, performing an element-by-element multiplication followed by a sum, resulting in a single output value. The outputs of each filter are then stacked to create a feature map that represents the activation of that filter on the input data. The aim of the convolutional layer is to capture and extract from the input data the features relevant to the task in hand, such as edge de- tection, texture recognition or object detection.

Pooling layer
A pooling layer is generally used after a convolution layer to reduce the spatial dimensions of feature maps. To do this, the feature map is divided into small, non-overlapping regions, and the maximum, minimum or average value of each region is taken into account. This operation reduces the number of parameters in the network, makes the model less sensitive to small variations in the input and avoids over-fitting.

Classification layer
The classification layer is generally located at the end of the CNN and takes as input the flattened feature maps of the previous layers. It consists of a set of neurons, each of which is connected to all the neurons in the previous layer. The aim of the fully connected layer is to use the features extracted from the previous layers to classify the input data. This layer is often followed by a softmax activation function, which converts the output of each neuron into a probability distribution of possible classes. The class with the highest probability is then chosen as the predicted output.

2.2 Architectures used

The five different models we've used all have their own architecture spe- cificities. We will briefly summarize these.

VGG-16
The VGG-16 consists of 16 layers, including 13 convolutional layers and 3 fully connected layers. Each convolutional layer has 3x3 filters and a step size of 1, followed by a rectified linear activation function (Re- LU) and a maximum pooling layer of 2x2. Fully connected layers comprise 4096 neurons each and are followed by a softmax activation function for classification.

EfficientNetV2_M
EfficientNetV2_M is a medium-sized model that uses a combination of convolutional layers and blocks. mobile inverted bottleneck blocks. The model consists of layers of rods, several inverted bottleneck blocks of varying widths and depths, and a head layer for classification. The inverted bottleneck blocks use depth and point convolutions to reduce the number of parameters and increase efficiency.

EfficientNet-B0
EfficientNet-B0 is the smallest model in the Effi- cientNet family, designed for resource-constrained environments. It uses a similar architecture to Efficient- NetV2_M, but with fewer layers and smaller filters. The model consists of stem layers, multiple inverted bottleneck blocks with variable widths and depths, and a top layer for classification.

ResNet-50
ResNet-50 is a deep CNN architecture that introduces the concept of residual connections. The model consists of stem layers, multiple residual blocks with a variable number of convolutional layers, and a main layer for classification. Residual connections facilitate the flow of information in the network and alleviate the problem of gradient disappearance.

DenseNet-121
DenseNet121 is a CNN architecture that uses dense blocks, which concatenate the outputs of all previous layers as inputs to the current layer. The model consists of stem layers, mul- tiple dense blocks with a variable number of convolutional layers, and a main layer for classification. Dense blocks enable more efficient use of para- meters and gradients, and promote the reuse of ca- racteristics.

2.3 Fine-tuning models

The five models listed in the previous subsection are deep learning models and therefore require significant computational resources in order to train them. For this reason, we decided to use pre-trained versions of these models to reduce the resources required for training. So we took models trained on the ImageNet dataset and fitted them to the CIFAR-10 and CIFAR-100 datasets. Adjusting a pre-trained CNN on a new dataset mainly consists in adapting the size of the output layer so that it correctly reflects the number of classes in our dataset. Ima- geNet is a dataset composed of 1000 classes. However, CIFAR-10 and CIFAR-100 are composed of 10 and 100 classes respectively. To solve this problem, we simply modify the output size of our models according to the dataset we want to train them on. As shown in Figure 1, we modify only the classification layer and then train the entire model.

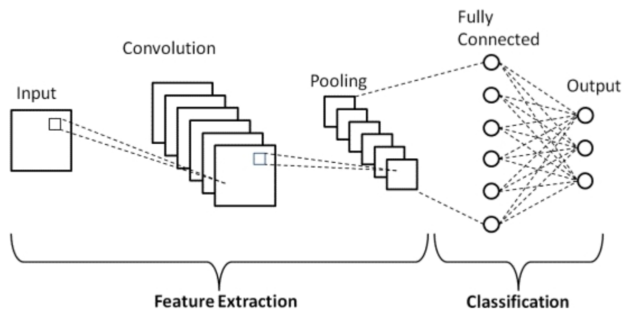


Figure 1: Typical architecture of a convolutional neural network

3 Experiences

This section describes the experiments carried out and the drive configuration used.

3.1 Experiments and hyperparameters

As part of the project, a number of experiments were carried out before deciding on the drive configuration we would choose for our five different CNNs. By
all our exploratory experiments w e r e c o n d u c t e d
s o l e l y o n t h e
conducted solely on the VGG-16 model, before deciding if it were necessary to reproduce it on other models.

Initially, we decided to use data augmentation techniques to achieve a very high level of accuracy. However, this considerably increases the training time required.
for merely a slight improvement in performance, we decided not to continue with this experiment.

The second experiment carried out was to try to fully implement VGG-16 and train it with randomly initialized weights to see how much training time was required. This experiment took a long time and was abandoned.

Hyperparameters

To compare the performance of our different models, we chose to use the same hyperparameters for all of them. We could have chosen for each CNN the hyperparameters with which we obtain the best results, but this would have required more time as we would have had to find the optimal configuration for each. So Table 1 shows the values for each hyperparameter.

Hyperparameter	Value
Batch size	32
Number of epochs	5
Apprenticeship rate	0.001
Momentum	0.9
Weight decay	0.0005

Table 1: Hyperparameters used

Learning algorithm and loss function

We decided to use the stochastic gradient descent algorithm to train our model. As for

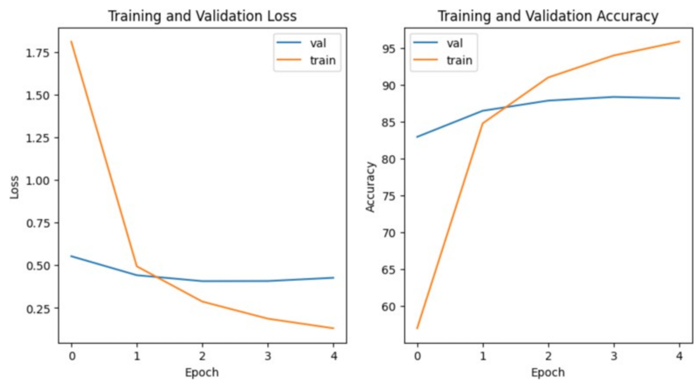


Figure 2: Loss and accuracy during Efficient- training NetV2_M on CIFAR-100

For the loss function, we used the cross-entropy loss function.

4 Results

This section presents the performance of our five different CNNs on the CIFAR-10 and CI- FAR-100 datasets.

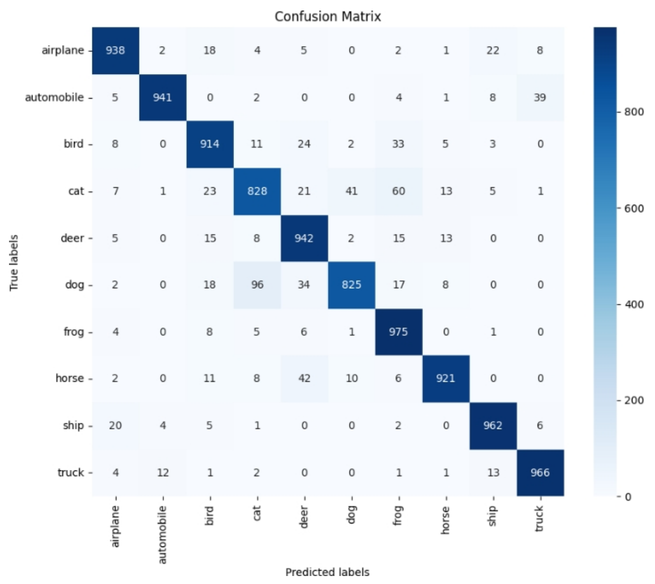
CNN	CIFAR-10 (%)	CIFAR-100 (%)
VGG-16	92.1	73.7
EfficientNetV2_M	**97.7**	**87.6**
EfficientNet-B0	96.1	81.7
ResNet-50	96.1	82.3
DenseNet-121	96.4	82.1

Table 2: Accuracy of CNNs on CIFAR-10 and CIFAR-100

In Table 2, the results show that all five CNN models are able to classify images with high accuracy on both CIFAR datasets. Unsurprisingly, all models perform better on CIFAR-10 than on CIFAR-100. This is because CIFAR-100 is a larger dataset and has 10 times more image classes than CIFAR-10.

The EfficientNetV2_M model achieves the best results on both data sets, with an accuracy of 97.7% on CIFAR-10 and 87.6% on CI- FAR-100. EfficientNet-B0, ResNet-50 and DenseNet-121 also achieve similar results, with scores above 96% on CIFAR-10 and above 81% on CIFAR-100. The VGG-16 model is the least successful of the five, with an accuracy of 92.1% on CIFAR-10 and 73.7% on CIFAR-100.

Figure 2 shows the average accuracy and loss on the training and validation datasets for EfficientNetV2_M and CIFAR-100. From this graph we can see that from a certain point onwards (epoch 3 here) the loss on the validation set no longer decreases, but rather starts to increase. It is at this point that the model is the best, so we made sure to save the best model to evaluate the perfor- mance on the test set. Of course, we also performed this procedure for the other CNNs.



	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane	938	2	18	4	5	0	2	1	22	8
automobile	5	941	0	2	0	0	4	1	8	39
bird	8	0	914	11	24	2	33	5	3	0
cat	7	1	23	828	21	41	60	13	5	1
deer	5	0	15	8	942	2	15	13	0	0
dog	2	0	18	96	34	825	17	8	0	0
frog	4	0	8	5	6	1	975	0	1	0
horse	2	0	11	8	42	10	6	921	0	0
ship	20	4	5	1	0	0	2	0	962	6
truck	4	12	1	2	0	0	1	1	13	966

Figure 3: Confusion matrix for VGG-16 on CIFAR-10

Figure 3 shows a confusion matrix of predictions made by VGG-16 on the CI- FAR-10 dataset. This representation is useful for identifying anomalies in the results. In this case, nothing abnormal has occurred, but we can see that the model is relatively wrong between cats and dogs. It tends t o predict cats more than dogs, and vice versa.

Figures 2 and 3 are examples, but these analyses were carried out on all the other CNNs too, and the graphs are all available at the following address: https://github.com/sanmarsimon/CNNs-CIFAR10-100

5 Discussion

This project has enabled us to study the operation of several different convolutional neural networks in depth and detail. CNNs are a mainstay of computer vision, and are the subject of ongoing research, with new architectures emerging all the time. Increasingly, these new architectures use a large number of parameters, and can require significant resources to train. That's why fine-tuning pre-trained models is such an interesting solution, as we've seen it can achieve very good results with much less training time. Our experiments have shown that the most recent of the five models (EfficientNetV2_M) achieves the best results.

In the future it might be interesting to compare the results obtained during this project to new experiments implementing transfer learning. Transfer learning involves simply training the classification layers of a pre-trained model.

References

[He et al., 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun. *Deep Residual Learning for Image Re- cognition.* https://arxiv.org/abs/1512.03385

[Tan and Le, 2019] Mingxing Tan and Quoc V. Le. *Efficient- Net: Rethinking Model Scaling for Convolutional Neu- ral Network.* https://arxiv.org/abs/1905.11946

[Kumar et al., 2021] Kumar, K., Mhetre, A., Ratnaparkhi, G.S., Kamat, S.S. (2021). A Superfamily-wide Activity Atlas of Serine Hydrolases in Drosophila melanogaster. Biochemistry 60(16): 1312--1324.

Kadam, S. S., Adamuthe, A. C., & Patil, A. B. (2020). CNN model for image classification on MNIST and fashion-MNIST dataset. *Journal of scientific research*, *64*(2), 374-384.