

Deep Learning Architectures for Image Classification

Michael Polonio

Introduction

As we progress in the information age computer vision becomes ever more popular. Image classification, a fundamental task of computer vision, has applications in many fields such as the automotive industry, medical field, manufacturing, and many more. Since the vast amount of image data we obtain from cameras and sensors is unstructured, we depend on advanced techniques such as machine learning and deep learning algorithms to analyze the images efficiently (Boesch, 2022). Image classification is the process of predicting what the image is of and can be done by analyzing each of its pixels individually. This problem is well-suited towards deep learning. Several deep learning techniques such as the Multilayer Perceptron, Convolutional Neural Network, and Transfer Learning will be trained for image classification and evaluated by their accuracy in this report.

Problem Statement

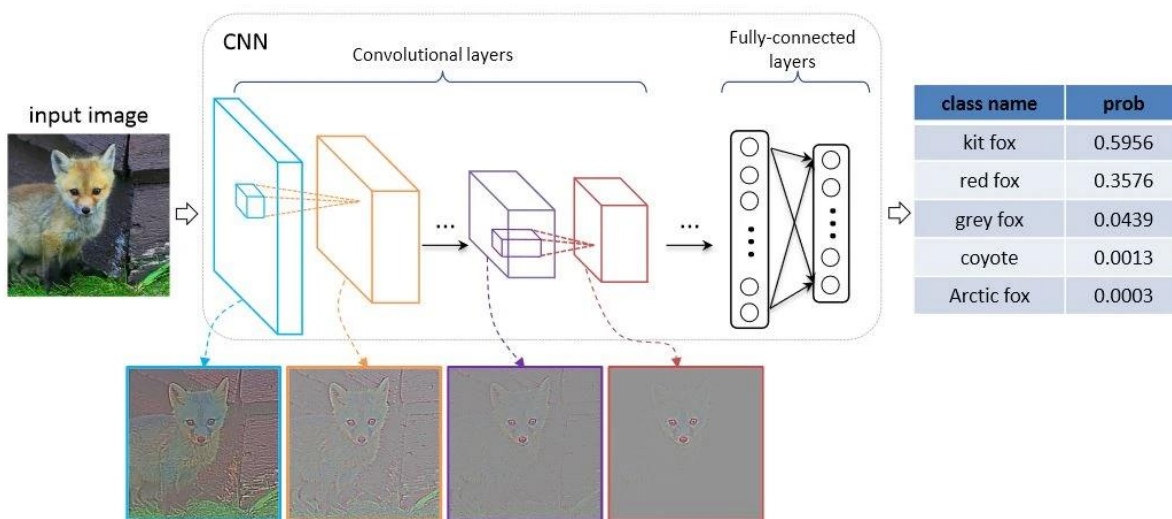
Can using *transfer learning* greatly increase the accuracy of an image classification model?

What are the performance differences between deep learning models of varying architecture and complexity?

Background

The power of neural networks comes from their ability to learn the representation in your training data and how to best relate it to the output variable that you want to predict (Brownlee, 2020). A *Multilayer Perceptron* is among the most basic of deep learning architectures. It consists of an input layer fully connected with weights to hidden layers consisting of neurons containing an activation function, and an output layer. The *Convolutional Neural Network* is considerably more complex than the MLP. Instead of simply consisting of input/output and hidden layers the CNN has three types of layers: convolutional layer, pooling layer, and the fully connected (FC) layer. Each layer contains different components whose explanations are out of the scope of this paper. *Transfer Learning* is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task (Brownlee, 2017). I will be using both the VGG-19 and the ResNet50 pre-trained models for transfer learning.

VGG Transfer Learning Architecture:

































































































Source: [Visualizing and Comparing AlexNet and VGG using Deconvolutional Layers \(icmlviz.github.io\)](https://icmlviz.github.io/)

Data

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images. This dataset is available to download straight from Keras.

Here are the classes in the dataset, as well as 10 random images from each:

airplane										
automobile										
bird										
cat										
deer										
dog										
frog										
horse										
ship										
truck										

Alex Krizhevsky (2013-2017). [CIFAR-10 and CIFAR-100 datasets \(toronto.edu\)](https://www.cs.toronto.edu/~kriz/cifar.html)

Methods

For this analysis I used Python and the Keras library for deep learning. Normalization was performed on all data values on a scale from 0 to 1. Four different models were constructed and

hyperparameter tuning was performed. The models were a Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and two transfer learning models (VGG-19 & ResNet50).

MLP: First flattening was performed to convert the matrix of pixels into a single array then was sent to the first hidden layer which consisted of 128 neurons and a sigmoid activation function. Then a 20% dropout was applied to avoid overfitting. The output layer had 10 neurons representing the different label outcomes. This model is compiled with a Stochastic Gradient Descent optimizer and a categorical crossentropy loss function.

CNN: First layer was a 2D convolutional layer with 32 filters, a 3x3 kernel size, and a ReLU activation function (along with all other layers). Then a 30% dropout was applied to avoid overfitting. Next was another 2D convolutional layer with 32 filters, a 3x3 kernel size, and a max pooling layer with a 2x2 pool size. Then another two convolutional layers this time with 64 filters, one with a 30% dropout and the next with a max pooling layer. Then two convolutional layers with 128 filters each followed by another max pooling layer. The the matrix is flattened and fed through a conventional MLP hidden layer with 128 nodes, the the output layer with 10 nodes representing the different outcomes. This model is compiled with a Stochastic Gradient Descent optimizer and a categorical crossentropy loss function.

VGG-19 Transfer Learning: The first part of this model was the VGG-19 pre-trained model. The VGG-19 is a CNN with 16 layers and trained on 14 million images belonging to nearly 1000 classes (Boesch). The data was trained on VGG then flattened and passed through a succession of hidden layers with 256, 128, 64 nodes with ReLU activation functions. Each layer has a dropout to avoid overfitting. The model was optimal at 5 epochs.

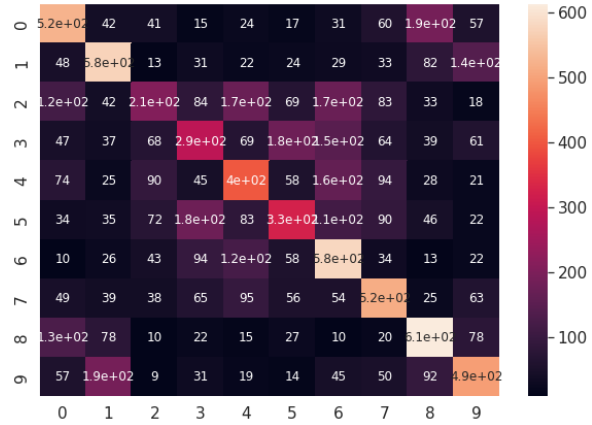
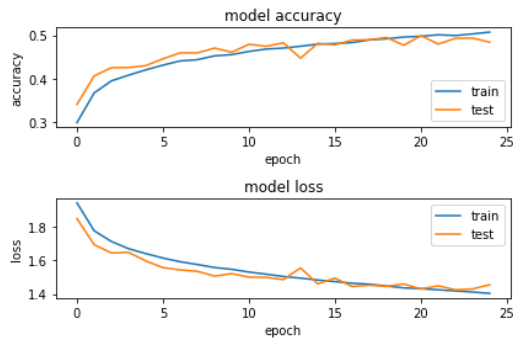
ResNet-50 Transfer Learning: The first part of this model was the ResdNet-50 pre-trained model. Resnet is a CNN that is 50 layers deep & trained on more than one million images. The data was trained on ResNet-50 then flattened and passed through two hidden layers with 64 and 128 nodes. Both ReLU activation function and a dropout of 30%. This model was compiled with an Adam optimizer. This model was optimal at 50 epochs.

Results

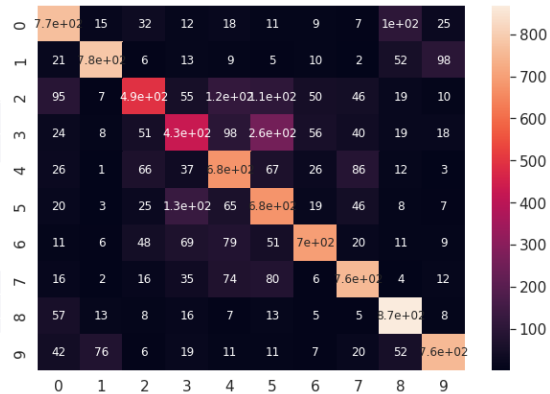
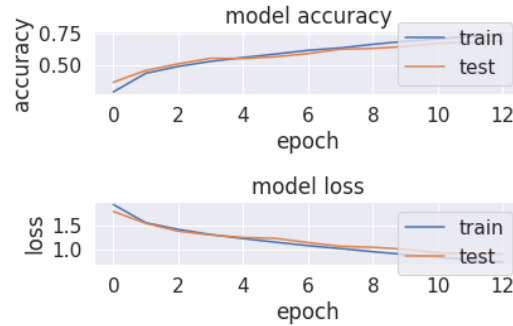
Out of the four deep learning architectures that were used for image classification of the CIFAR-10 images, the VGG-19 transfer model outperformed the other with an accuracy of 85%. The CNN outperformed the more robust ResNet-50 transfer learning model achieving an accuracy of 74%. The ResNet-50 model performed worse than the less complex CNN, possibly indicating ResNet is overkill for this problem. The least complex Multilayer Perceptron performed with the lowest at an accuracy of 50%.

Model	Accuracy
Multilayer Perceptron	50%
Convolutional Neural Network	74%
VGG-19 Transfer Learning	85%
ResNet-50 Transfer Learning	71%

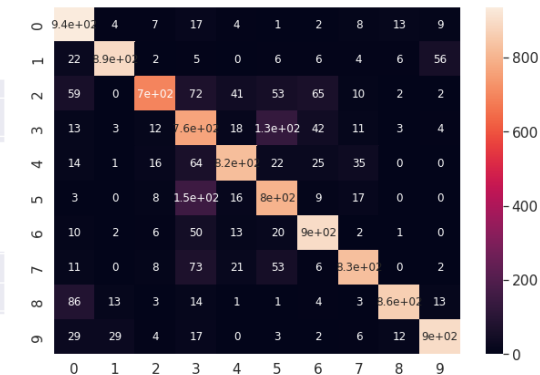
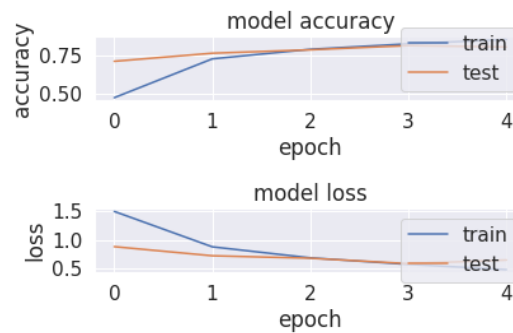
MLP (50%):



CNN (74%):



VGG-19 (85%):



ResNet-50 (71%):



Conclusion

The results illustrated effectively that varying architecture complexities can have a substantial impact on performance. An additional 11% accuracy was gained by the VGG-19 transfer learning model over the CNN. The various architectures yielded a wide range of results with a difference of up to 35% (VGG-19/MLP). In transfer learning, multiple different models should be trained. Just because a model is more robust does not mean it will perform better. The ResNet-50 model was significantly deeper than the VGG-19 but still was outperformed by 14%.

I believe there can be commercial or experimental applications of the VGG-19 model built for this report due to its high accuracy of 85%. I surmise that an equal or greater accuracy can be obtained by image classification in a wide variety of class labels. I think more models of this architecture should be constructed with the other available pre-trained models for transfer learning to see if a higher accuracy can be obtained.

References

Boesch, Gaudenz (2020). A Complete Guide to Image Classification in 2022. [A Complete Guide to Image Classification in 2022 - viso.ai](#)

Brownlee, Jason (2016). Crash Course On Multi-Layer Perceptron Neural Networks. [Crash Course On Multi-Layer Perceptron Neural Networks \(machinelearningmastery.com\)](#)

Brownlee, Jason (2017). A Gentle Introduction to Transfer Learning for Deep Learning. [A Gentle Introduction to Transfer Learning for Deep Learning \(machinelearningmastery.com\)](#)

Yu, W., Yang, K., Bai, Y., Xiao, T., Yao, H., & Rui, Y. (2016, June). Visualizing and comparing AlexNet and VGG using deconvolutional layers. In *Proceedings of the 33 rd International Conference on Machine Learning*. [Visualizing and Comparing AlexNet and VGG using Deconvolutional Layers \(icmlviz.github.io\)](#)

Boesch, Gaudenz. VGG Very Deep Convolutional Networks (VGGNet) – What you need to know. [VGG Very Deep Convolutional Networks \(VGGNet\) - What you need to know - viso.ai](#)

[CIFAR-10 and CIFAR-100 datasets \(toronto.edu\)](#)

Plotka, Szymon (2018). CIFAR-10 CLASSIFICATION USING KERAS TUTORIAL. [CIFAR-10 classification using Keras Tutorial - Ermlab Software](#)

Krizhevsky, A., & Hinton, G. (2009). Learning multiple layers of features from tiny images.