Machine Learning Course Ex - 3

Prof Lior Rokach

BGU

Ofir Arbili 034121392

July 17th 2022

Introduction

The paper describe the solution for Assignment 3

There were two parts to the assignment:

1. An additional layer is added to an Artificial Neural Network implementation, and the results are compared with the original implementation and an implementation using the NN package (keras in my solution).
2. To identify flowers images using pretrained CNNs (at least 2), using the Oxford db.

Part-1

Implementation

The code was extended to address two hidden layers (instead of one hidden layer) by changing the following functions: forward(), backward() and also changed the init() to allow selection between 1 and 2 layers.

Forward function:
The original function renamed to forward_one_layer, a new 2-layers function is added and called forward_two_layers(). This function is a straight-forward calculation of the NN, which involves multiplying inputs with weights, adding baises, and using the results as the input for activation function.

Backward function:
The original function renamed to backward_one_layer, a new 2-layers function s added and called backward_two_layers(). In this function, the derivatives of the loss are calculated for each backward "step" (each neuron in each layer). Each layer's weights and biases are updated based on the results of the backward_two_layers function.

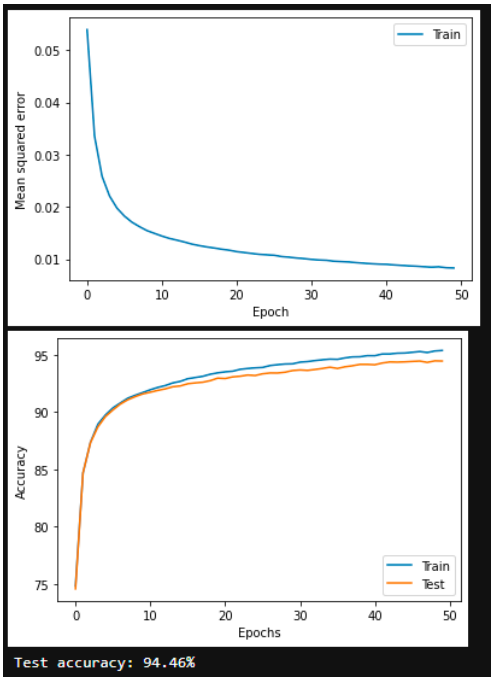
Additional new functions:
In this function, a prediction is made based on input, the related forward function is called accordingly.
Print_results_charts - (in utils) - print the MSE and accuracy chart
Plot_roc_curve - (in utils) - print the ROC AUC curve

Results

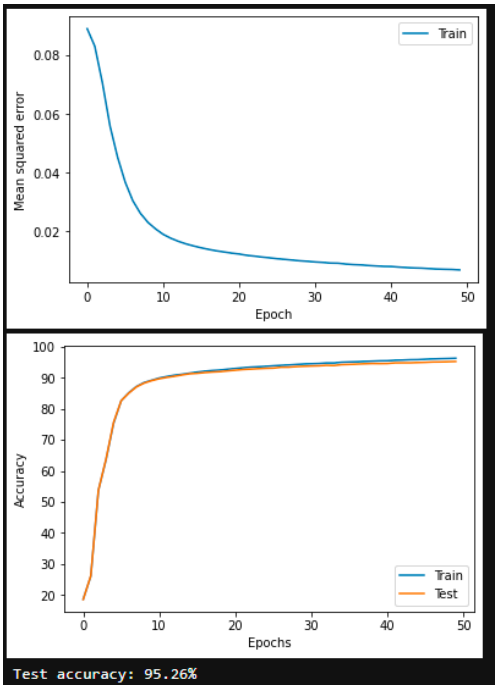
Network Type	**Accuracy**	**AUC-Macro**
1 layer NN	0.9446	0.9925
2 layers NN	0.9526	0.9945
Keras	0.9693	0.9988

- **MSE and Accuracy charts:**

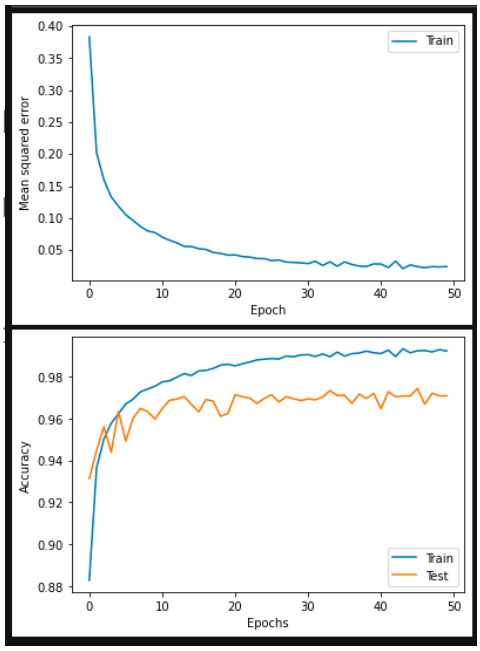
1-Layer NN



2-Layers NN

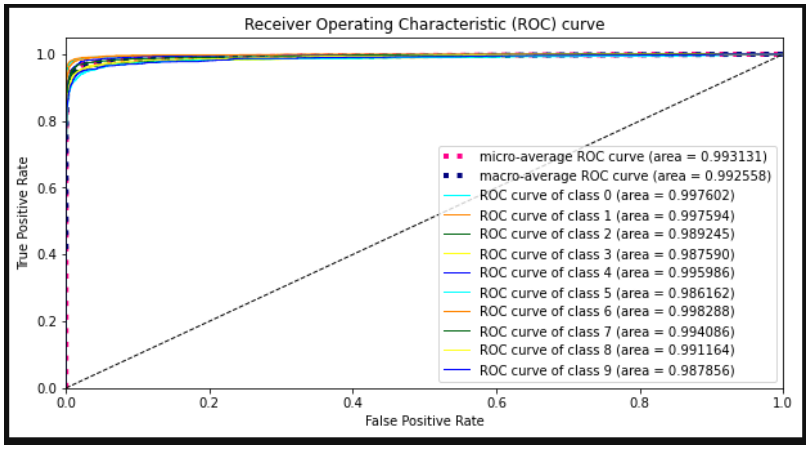


Keras NN

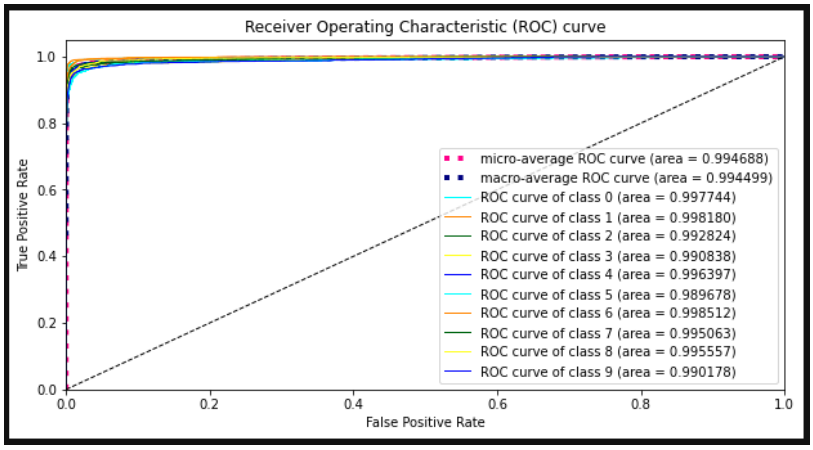


- **ROC Curve:**

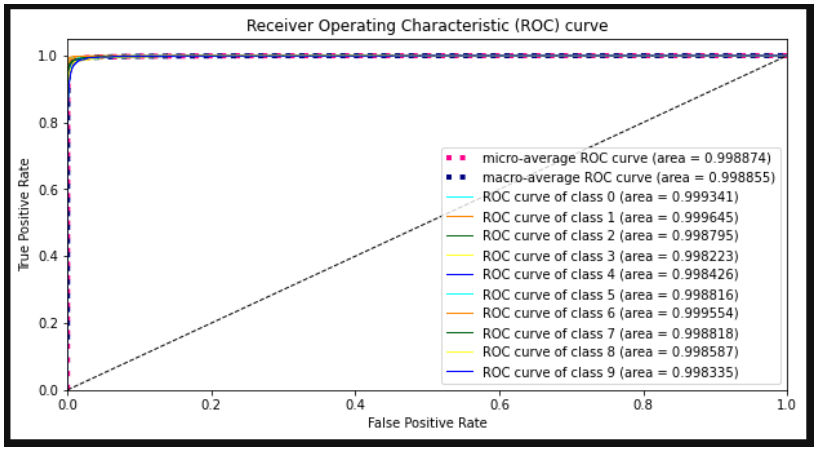
1-Layer NN



2-Layers NN



Keras



Part-2

The selected pretrained CNNs are resnet101 and EfficientNet

Resnet101

The Resnet101 consists of 101 layers of convolutional neural networks. You can load a pretrained version of the network trained on more than a million images from the ImageNet database. It can classify images into 1000 object categories, such as keyboards, mice, pencils, and many animals.

The resnet family includes 18,34,50,101 and 152 layers network. I used the 101 layers. The following table describe the structure of each network.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

The main advantages of ResNet is while it increases network depth, it avoids negative outcomes. As a result, the depth can be increased but the training is faster and the accuracy is higher.

EfficientNet

EfficientNet is an image classification model family. It was first described in “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks”. In this work I tested EfficientNet-B3, EfficientNet-B1ns, EfficientNet-WideSE-B4ns. In general, the EfficientNet models achieve both higher accuracy and better efficiency over existing CNNs, reducing parameter size and FLOPS by an order of magnitude.

The baseline b0 network architecture:

Preprocess:

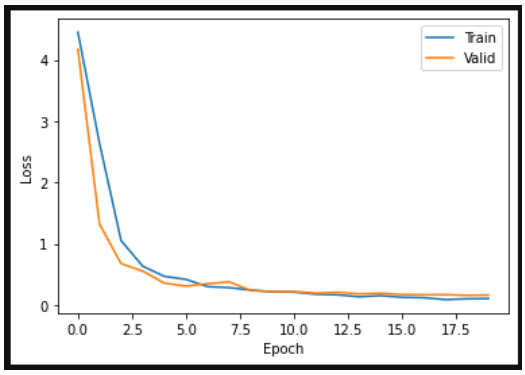
Image preprocess:

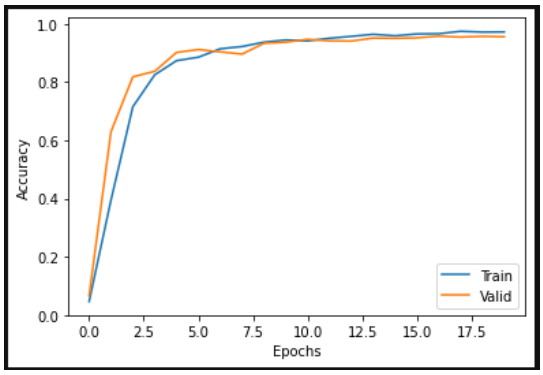
- Resized
- Horizontal Flip
- Rotation
- GaussianBlur

Results:

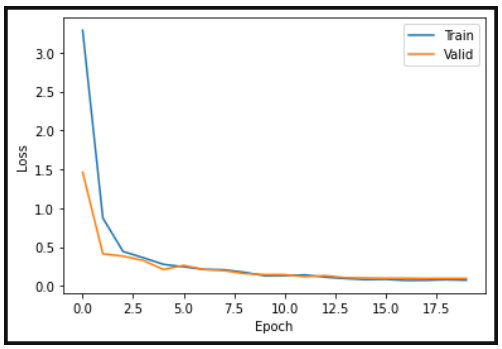
Model	Valid Loss	Best Valid Accuracy	Test Accuracy
ResNet101	0.1654	0.9722	0.9692
EfficientNet_b1_ns	0..0964	0.9771	0.9746
efficientnet_b3	0.0.1076	0.9722	0.9741
efficientnet_b4_ns	0..0547	0.9800	0.9814

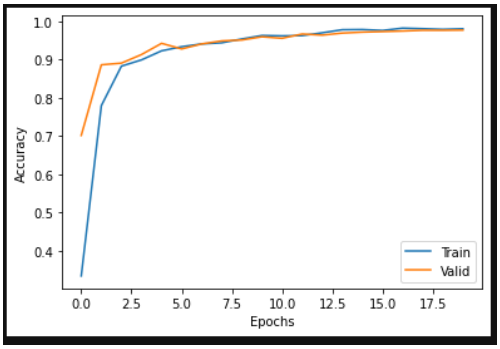
ResNet101



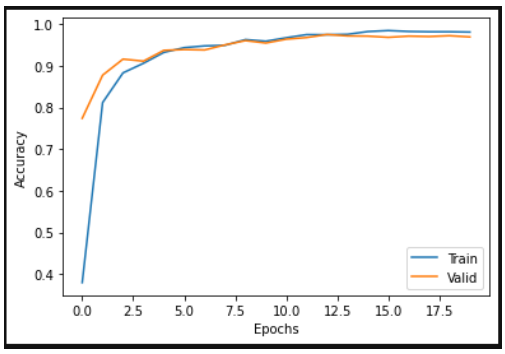


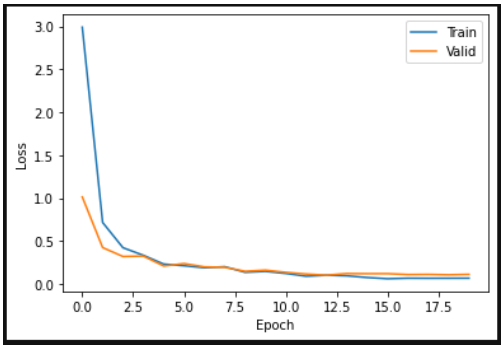
EfficientNet_b1_ns





EfficientNet_b3





EfficientNet_b4_ns