

Project 2 - Computer Vision

Introduction

In this exercise, you're going to train an image classification algorithm using the CIFAR10 dataset. This dataset contains 60,000 small images, each 32x32 pixels, categorized into 10 different classes (such as airplane, automobile, etc.). The compact size of these images enables fast experimentation with various Computer Vision principles without the need for a GPU. We will use a subset of the dataset for faster experimentation.

Description

The code provided trains an image classifier using a subset of the CIFAR10 dataset and the MobileNet architecture as its foundation. It runs for 5 epochs, with every layer except the final one being set to non-trainable (frozen).

Installation

Create a new Anaconda environment with Python 3.8.19. Use pip to install all required libraries from requirements.txt

1. Create new anaconda environment -
2. Activate the environment - `conda activate myenv`
3. **pip install -r requirements.txt**

Exercise

1. Create a function that draws a plot using the variable *'history'*.
 - Draw a plot of train/validation accuracy per epoch using matplotlib. Draw the train accuracy in **green** and validation in **blue**.
 - Draw a plot of train/validation loss per epoch using matplotlib. Draw the train loss in **red** and validation loss in **yellow**.
2. Based on the plots,
 - A. Do you think that training for more epochs will improve the results? Explain.
 - B. Do you think that increasing the dataset will improve the results? Explain
3. For each of the following (A,B,C,D,E), write down the test accuracy, and training time. **Hint :** (<https://docs.python.org/3/library/timeit.html>).

For each item,

- Explain (in one sentence) why the accuracy improved. .
- Explain (in one sentence) why training is slower, faster or hasn't changed.

Note: Changes are cumulative, for example when doing part B, keep the layers unfrozen (part A).


First, write down the basic test accuracy and inference time in the first row.

- A. Unfreeze the layers (*base_model.trainable*) so that the whole network can learn. Hint : https://keras.io/guides/transfer_learning/
- B. Change the learning rate from the default *learning_rate* to 0.01. Hint: <https://keras.io/api/optimizers/>
- C. Add a dropout layer to the DNN, with a dropout rate of 0.5, right after global average pooling. Hint: https://keras.io/api/layers/regularization_layers/dropout/
- D. Add random flip augmentation to the DNN. Hint: https://keras.io/api/layers/preprocessing_layers/image_augmentation/random_flip/
- E. Change MobileNet to ResNet18 and retrain for 5 epochs. Calculate the average latency of MobileNet vs ResNet18 using the test set. Use (https://github.com/qubvel/classification_models) to get ResNet18. Use “pip install” and update “requirements.txt”. Add a parameter to *create_model* function for easier switching between the models.
- Hints:
- (<https://docs.python.org/3/library/timeit.html> - execution time)

Advanced Section - Extra Points

In this section, we will build from scratch a full computer vision solution, including dataset collection, tagging, training and inference. Let's build a system that can count Israeli coins. The input to the system would be an image, which is a photo of several coins and the output is the total amount in NIS.

Example:

	Result : 17 NIS
---	------------------------

For simplicity's sake, assume that there are no overlaps between the coins and the angle of view is reasonable.

We will focus on the following coins:

Coin	Class Name
1 Shekel	One
2 Shekels (Shnekel)	Two
5 Shekels	Five
10 Shekels	Ten

1. Collect using your smartphone a dataset of at least 50 images with different varieties of coins, backgrounds, angles, illuminations. Split the dataset into train and test.
2. Tag the coins bounding boxes and classes using the class names given above. You can use Vertex.ai [datasets tagging tool](#), or any other object detection labeling tool .
 - a. You can share the tagged dataset with your friends to increase the amount of images.
3. Train an object detector that returns one of the four classes, using your own dataset
 - a. Try fine-tuning an existing object detection algorithm. You can use huggingface
4. Write a code that calculates the total amount of money in an image
5. Write a metric that calculates the accuracy by comparing the normalized absolute difference between the amount of money in an image and the predicted amount.
 - a. For example:
 - i. Predicted - 20 NIS
 - ii. Ground Truth - 25 NIS
 - iii. Accuracy per image = $1 - \frac{|25-20|}{25} = 0.8$, which is 80% accuracy
 - b. Write a metric that calculates the average accuracy over a dataset of N images.
6. What is your accuracy on the test set?

Submission:

Provide the results, including the plots and your explanations in a Google Doc. Use the attached template. Also provide your final Python code.

Citations:

- Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009.

- Deep Residual Learning for Image Recognition, Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 2015
- MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam