Last updated on Sunday, September 18, 2022

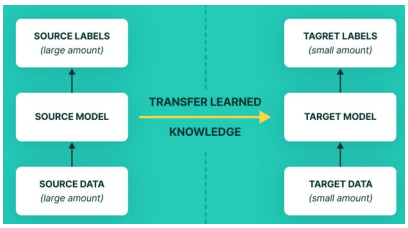
Transfer Learning Assignment

Key information

- Submission due at the end of **Week 13** (Sunday 30 October, 23.59pm)
- Submit your work via Blackboard
- Recommended group size: three people per submission. Smaller groups are allowed (1 or 2 people OK, but completion of the same tasks is required).

Overview

Transfer Learning is a machine learning method where we reuse a model trained on a first dataset called the *source dataset* as the starting point for training a model on a second dataset called the *target dataset*. Generally, the source dataset is a large dataset like *ImageNet* and the target dataset is a much smaller dataset relevant to a new application domain.



The aim of this assignment is to build a flower classifier using transfer learning on a neural network trained on the ImageNet dataset

Transfer Learning

In practice, very few people train an entire Convolutional Network (ConvNet) from scratch (with weights initialized randomly), because it is relatively rare to have a dataset of sufficient size. Instead, it is common to pretrain a ConvNet on a very large dataset (e.g. ImageNet, which contains 1.2 million images with 1000 categories), and then use the ConvNet either as an initialization or a fixed feature extractor for the task of interest. In other words, transfer learning is usually done for tasks where your dataset has too little data to train a full-scale model from scratch.

The most common incarnation of transfer learning in the context of deep learning follows the workflow below:

- Take a slice of layers from a previously trained model.
- Freeze their weights, so as to avoid destroying any of the information they contain during future training rounds on your new dataset.
- Add some new, trainable layers on top of the frozen layers. They will learn to turn the old features into predictions on a new dataset. "Dense Layer"
- Train the new layers on your new dataset.

A last, optional step, is *fine-tuning*, which consists of unfreezing the entire model you obtained above (or part of it), and re-training it on the new data with a very low learning rate. This can potentially achieve meaningful improvements, by incrementally adapting the pretrained features to the new data. **Fine-tuning is not required for this assignment!**

Approach

Recall that in a convolutional neural network, lower convolutional layers[1] capture low-level image features, e.g. edges or corners, while higher convolutional layers have a larger receptive field and capture more and more complex details, such as body parts, faces, and other compositional features.

The final fully-connected layers are generally assumed to capture information that is relevant for solving common computer vision tasks. For example, the fully-connected layers of the *MobileNet* network can be interpreted as generic computer vision features that are relevant to classify an image into one of the 1000 object categories of the ImageNet dataset.

We choose *MobileNetV2* as the neural network model architecture because it is lightweight. MobileNetV2 has "only" 3.5 millions parameters! It is the smallest of the available pretrained networks in Keras[2].

When you perform transfer learning without fine-tuning, you can "accelerate" the training process of the last layers by creating an auxiliary dataset the following way: let us call **F(x)** the function computed by the frozen layers of the neural network, and let us call **N(x)** the function implemented by the new layers. Your new network implements the function **N(F(x))**. That is, **F** composed with **N**. Assume our dataset is **{(x1,t1), (x2,t2),…, (xm,tm)}**. Standard transfer learning sets the learning rate of the weights of **F** to zero and applies the backpropagation algorithm to the network **x→N(F(x))**. By precomputing the activation arrays **{F(x1),F(x2),…,F(xm)}**, we can create a dataset **{(F(x1),t1), (F(x2),t2),…, (F(xm),tm)}** for the network **z→N(z)**. The gain is that we don't have to recompute the activations **F(x)** when we apply the backpropagation algorithm directly to the network **z→N(z)**. F can be precomputed to speed up training, because it doesn't change

encoding unsigned int between 0-256

preprossing the images to be normalised to values between 0 and 1

1 Those closer to the input layer
2 See the table of available model at https://keras.io/api/applications/

read keras on transfer learning

https://keras.io/guides/transfer_learning/

Your tasks

should get more than 90% accuracy

1. [1 mark] Download *the small flower dataset* from Blackboard.
2. [1 mark] Using the *tf.keras.applications* module download a pretrained MobileNetV2 network.

Perform "standard" transfer learning

3. [3 marks] Replace the last layer of the downloaded neural network with a Dense layer of the appropriate shape for the 5 classes of the small flower dataset **{(x1,t1), (x2,t2),…, (xm,tm)}**.
4. [2 marks] Prepare your training, validation and test sets for the non-accelerated version of transfer learning.
5. [2 marks] Compile and train your model with an SGD[3] optimizer using the following parameters *learning_rate=0.01, momentum=0.0, nesterov=False.*
6. [2 marks] Plot the training and validation errors vs time as well as the training and validation accuracies. accuracies including precision and recall?
7. [2 marks] Experiment with 3 different orders of magnitude for the learning rate. Plot the results, draw conclusions.
8. [2 marks] With the best learning rate that you found in the previous task, add a non zero momentum to the training with the SGD optimizer (consider 3 values for the momentum). Report how your results change.

Repeat a subset of the above experiments for the "accelerated" version of transfer learning.

9. [3 marks] Prepare your training, validation and test sets. Those are based on **{(F(x1).t1), (F(x2),t2),…,(F(xm),tm)}**,
10. [2 marks] Perform Task 8 on the new dataset created in Task 9.

All your code should be submitted in **a single Python script file**. In this file, you should structure your code so that your experiments can be easily repeated by uncommenting function calls in the main block of your script. That is, your code should look like

```
if __name__ == "__main__":
    pass
    # task_1()
    # task_2()
    :
```

Deliverables

You should submit via Blackboard only two files

1. A **report** in **pdf** format **strictly limited to 8 pages in total** (be concise!) containing
   - a description of your investigation with results clearly presented in tables and figures.
   - a recommendation for the values of the investigated parameters
2. A Python script to reproduce your experiments

   The file should include the task functions required to repeat your experiments. Repeating your work should be possible by uncommenting function calls in the main block of your Python file.

3 The SGD class description can be found at https://keras.io/api/optimizers/sgd/

Marking Guide

- **Task completion:** 20 marks
- **Report:** 15 marks
    - Structure (sections, page numbers), grammar, no typos.
    - Clarity of explanations.
    - Figures and tables (use for explanations and to report performance).
- **Code quality**: 15 marks
    - Readability, meaningful variable names.
    - Proper use of TensorFlow/Keras, Numpy, Python idioms like dictionaries and list comprehension.
    - No unnecessary use of loops when array operators are available.
    - Header comments in classes and functions. In-line comments.
    - Function parameter documentation.
    - Reproducibility of the experiments; the marker should only have to uncomment some function calls to repeat your work (if needed).

Marking criteria

Levels of Achievement

15 Marks	12 Marks	9 Marks	6 Marks	1 Mark
+Report written at the highest professional standard with respect to spelling, grammar, formatting, structure, and language terminology.	+Report is very-well written and understandable throughout, with only a few insignificant presentation errors. +Recommendation derived from experimental results are clear	+The report is generally well-written and understandable but with a few small presentation errors that make one of two points unclear. +Clear figures and tables.	The report is readable but parts of the report are poorly-written, making some parts difficult to understand. +Use of sections with proper section titles.	The entire report is poorly-written and/or incomplete. +**The report is in pdf format.**

To get "i Marks", the report needs to satisfy all the positive items of the columns "j Marks" for all $j \leq i$. *For example, if your report is not in pdf format, you will not be awarded more than 1 mark.*

Levels of Achievement

[12-15] Marks	[10-11] Marks	[7-9] Marks	[4-6] Marks	[1-3] Mark
+Code is generic, well structured and easy to follow. For example, auxiliary functions help increase the clarity of the code.	+Proper use of data-structures. +No unnecessary loops. +Useful in-line comments. +Header comments are clear. The new functions can be unambiguously implemented by simply looking at their header comments.	+No magic numbers (that is, all numerical constants have been assigned to variables with meaningful names). +Each function parameter documented (including type and shape of parameters) +return values clearly documented	+Header comments for all new classes and functions. +Appropriate use of auxiliary functions.	Code is partially functional but gives headaches to the marker.

To get "i Marks", the report needs to satisfy all the positive items of the columns "j Marks" for all $j \leq i$.

Miscellaneous Remarks

- Do not underestimate the workload. Start early. You are strongly encouraged to ask questions during the practical sessions or use MS Teams (**don't forget to tag me if you want to make sure I read your post**).
- Don't forget to **list all the members of your group in the report and the code**!
- Only one person in your group should submit the assignment.

How many submissions can I make?

- You can make multiple submissions. Only the last one will be marked.

How do I find team-mates?

- Post a message in the Assignment 2 channel of the unit MS Teams.
- Make sure you discuss early workload with your team-mates. It is not uncommon to see groups starting late or not communicating regularly and eventually submitting separately bits of work.
- You can stay with the same team as for the first assignment. But, it is also fine to change teams for the second assignment.
- When marking the assignment, we only look at the names that appear in the report and in the code. We ignore the Blackboard groups when marking.