Homework 2: Deep Learning

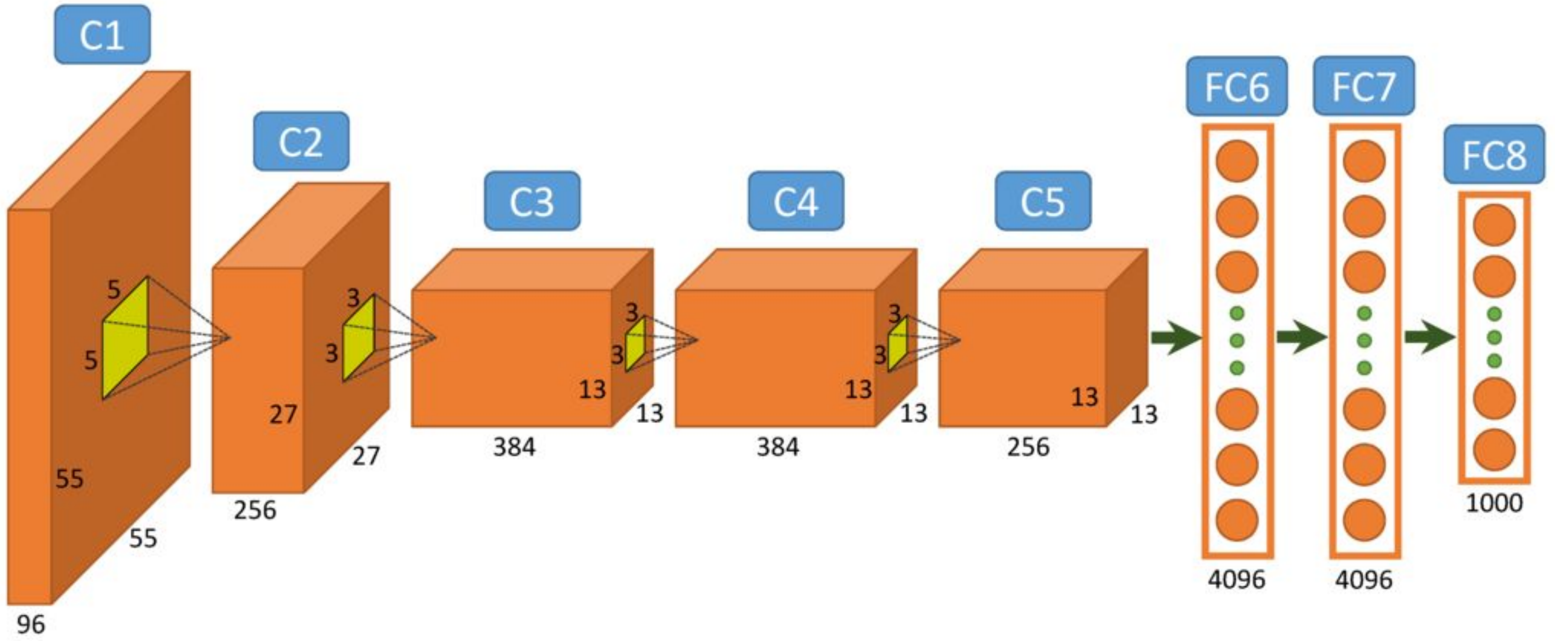
Antonio D'Innocente
dinnocente@diag.uniroma1.it

Overview

- You are going to train a Convolutional Neural Network for image classification
    - AlexNet on Caltech-101

- Start from a provided template code, and implement experiments

AlexNet (link)



Caltech-101 (link)

101 Classes (+ background)

Common Categories

GPUs

Deep Learning needs GPUs acceleration

We use Google Colab

- https://colab.research.google.com/

Code Template

- Starting code is available at:
  - https://colab.research.google.com/drive/1PhNPpklp9FbxJEtsZ8Jp9qXQa4aZDK5Y
  - Save a copy on your own drive “File -> Save a Copy on Drive” and start working on it
  - “Edit -> Notebook Settings”: Check that hardware acceleration is using GPU
  - If you find it more convenient, work and debug locally with CPU (remember to set the argument `DEVICE=’cpu’`), and upload working code when ready for training
- Dataset available via github repository:
  - https://github.com/MachineLearning2020/Homework2-Caltech101

Homework 2

0- First Step

A. Study code and data
   - Read the comments, understand what everything is doing
   - Pay extra attention to: Preprocessing, Datasets, Training, Testing
   - Also explore the data provided on github, and study the ImageFolder class of torchvision.datasets

B. Run the code
   - Training should take less than ten minutes
   - You have to stay connected

1- Data Preparation

- The dataset is created using the ImageFolder class
- Train/Test split are generated with an arbitrary criterion in the code using the Subset class
- However, train/test splits are actually provided in the github repository
- There is also a BACKGROUND folder in Caltech-101 that is used but shouldn’t be

1- Data Preparation

A. Use the provided train/test splits for training dataset and test dataset

B. Filter out the BACKGROUND class (which is incorrectly provided also in the splits)

There is not an unique way, however the most elegant solution is an ad-hoc dataset class which reads the split files. A template is provided in the github repository with the dataset

2- Training from scratch

- For training, we need a validation set for hyperparameter tuning and model selection
- In deep learning, it is often prohibitive to do k-fold cross validation, due to time required to complete trainings
- The most common procedure is to have a single validation set for both hyperparameter tuning and model selection

2- Training from scratch

A. Split the training set in training and validation sets
   - The validation set must be half the size of the training set
   - Be careful to not filter out entire classes from the sets!
   - Aim for half samples of each class in train and the other half in val

B. Implement model selection with validation
   - After each training epoch, evaluate (= test) the model on the validation set
   - Use only the best performing model on the validation set for testing

2- Training from scratch

The current implementation trains using SGD with momentum for 30 epochs with an initial learning rate of 0.001 and a decaying policy (STEP_SIZE) after 20 epochs

C. Experiment with at least two different sets of hyperparameters
- The first hyperparameter to optimize is the learning rate
- Low learning rate = the model converges too slowly (loss decreases slowly)
- High learning rate = the model converges fast but accuracy is sub-optimal
- Too high learning rate = the model diverges (loss increases)

3- Transfer Learning

- Deep Learning needs very large datasets to train good features
- Caltech-101 = very small dataset
- Solution: Transfer Learning (finetuning)
  - Use the weights learned by training on a large related dataset as a starting point for training on the small dataset

3- Transfer Learning

A. Load AlexNet with weights trained on the ImageNet dataset
   - (read the documentation)

B. Change the Normalize function of Data Preprocessing to Normalize using ImageNet's mean and standard deviation
   - (also search for this)

C. Run experiments with at least three different sets of hyperparameters

3- Transfer Learning

- When transfer learning, we can also “freeze” part of the network and only train certain layers (not always the best choice, but faster and can prevent overfitting in some cases)

With the best hyperparameters you obtained in the previous step:

D. Experiment by training only the fully connected layers

E. Experiment by training only the convolutional layers
   - Compare all results you obtained

4- Data Augmentation

- Deep Learning leverages huge amounts of data
- It is common to artificially increase the dataset size by applying label-preserving transformations to images
  - The image changes, but the label is the same
  - Prevents overfitting by increasing cardinality of data
- Examples of data augmentation include:
  - Cropping, Horizontal Flipping
- The current preprocessing for our input data is an Image Resize followed by a 224x244 central crop

4- Data Augmentation

A. Apply at least three different sets of preprocessing for training images
   - https://pytorch.org/docs/stable/torchvision/transforms.html
   - no need to use complex transformations, such as RandomAffine
   - if test data is very similar to training data, some data augmentation policies may worsen accuracy
   - some transforms (FiveCrop and TenCrop) are designed for evaluation only
   - if the loss keeps decreasing, increase training epochs or increase learning rate

Test-time data augmentation is less common in learning and research (but is used in competition). If you feel like it, try to integrate the TenCrop transform for test data (not trivial)

5- (Extra) Beyond AlexNet

- torchvision implements a lot of convolutional neural networks
    - https://pytorch.org/docs/stable/torchvision/models.html

- You can experiment with different models (like ResNets or VGGs)
    - some ConvNets need an input size higher than 224x224
    - Larger models consume GPU memory, you might need to decrease the batch size (a lot)
    - Larger models need much longer to train and Colab's K80s are not the fastest GPU around (freezing layers is an option)
    - Larger models are more accurate

Submission rules

- Suggested Deadline (more points): **16 December 23:59 (CET)**
- Hard Deadline: One week before the written exam
- Uploading: through “Portale della didattica”

Submit a zip named <YOUR_ID>_AIML_homework2.zip. The zip should contain two items:

- A pdf report describing data, your implementation choices, results and discussions
- Code