Computer Vision (CSE 40535 / 60535)

Practical 4:

CNN-based object classification

Adam Czajka
Department of Computer Science and Engineering
University of Notre Dame, USA

Fall 2020

Training of CNNs

Transfer learning

- Training CNN from scratch requires datasets of sufficient size and large computational resources
- Practice:
  - find a network pre-trained on a large dataset
  - use the network as a fixed feature extractor
  - or tune the (selected) network weights to your needs
- Example pre-trained models used in ILSVRC (ImageNet Large Scale Visual Recognition Competition)
  - ResNet (He 2016, the winner of ILSVRC 2015)
  - GoogLeNet (Szegedy 2015, the winner of ILSVRC 2014)
  - VGGNet (Simonyan 2014, the runner-up in ILSVRC 2014)
  - AlexNet (Krizhevsky 2012, the winner of ILSVRC 2012)

Training of CNNs

Transfer learning – variants

- Fixed feature extractor
  - take the pre-trained model and remove last FC layers
  - apply and train the classifier of your choice (multilayer perceptron, SVM) for multidimensional feature vectors (CNN codes) calculated for your new samples
- Fine-tuning of the network
  - fine-tune all the layers
  - keep first layers fixed and fine-tune the last layers

Training of CNNs

Transfer learning – practical scenarios

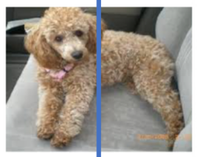
		Similarity between datasets (new vs original)	
		large	small
Size of the new dataset	small		
	large		

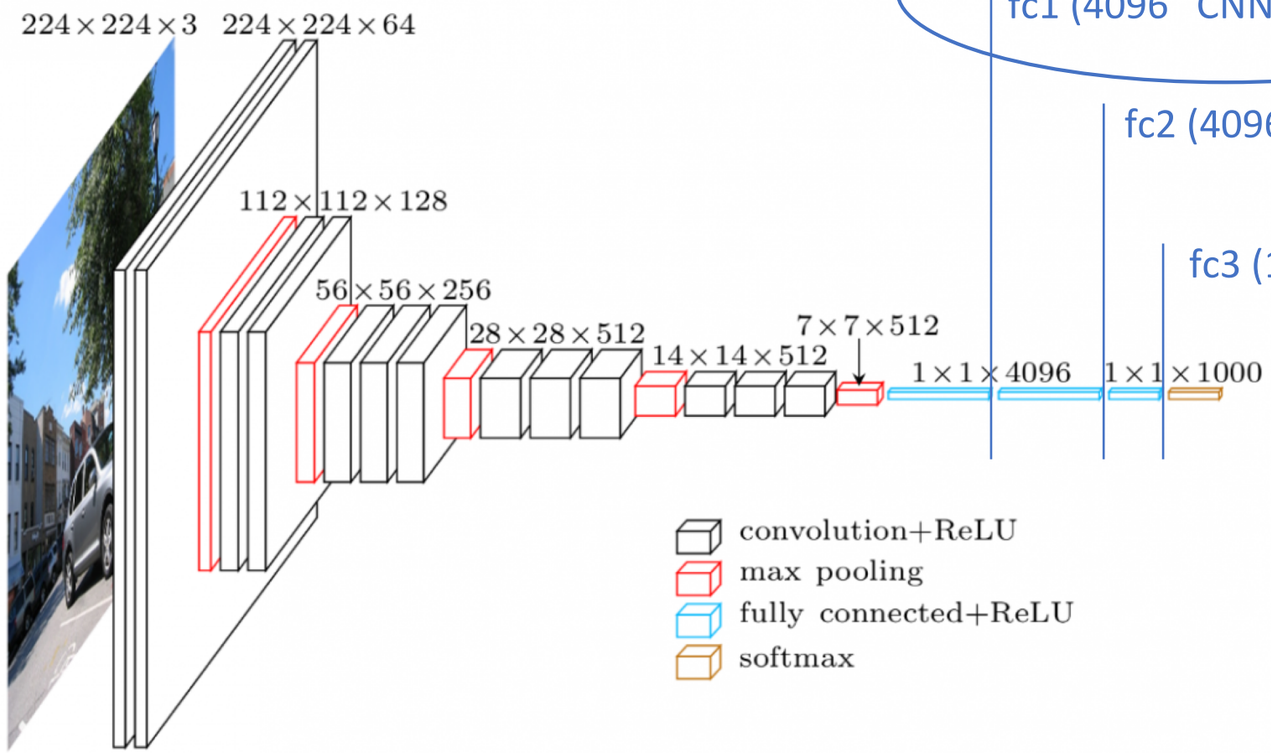
Practical 4: CNN-based object classification

What we are doing today?

VGG-16 trained on ImageNet (1000 classes)





fc1 (4096 “CNN codes”)

fc2 (4096 “CNN codes”)

fc3 (1000)

training on the CNN codes

SVM



New objects that we want to classify that **were not included** into ImageNet (CALTECH database)

Practical 4: Tasks

- Task 1 (0.5 point)
  - Select three classes corresponding to the objects you brought to the class
  - Train **linear SVM** using **fc1** CNN codes and try to recognize all your objects **presenting them to the camera.** Are the objects classified correctly?
- Task 2 (2 points)
  - Experiment with the **level of the CNN codes** (fc1 vs fc2) and **SVM kernel** ('linear', 'rbf', 'poly') for these three classes
  - Which configuration is the best? How would you explain why this selected configuration works best?
- Task 3 (0.5 point)
  - For the best configuration found in task 2, train your SVM for the **extended set of CALTECH classes** (10 or more, including your 3 classes selected earlier).
  - Does your solution still perform well? What is the reason for a different performance?
- Task 4 (mandatory for 60000-level attendees; optional for others)
  - Instead of handcrafting the best hyperparameters (SVM kernel and its parameters, embeddings level: "fc1" vs "fc2"), implement a simple grid search-based algorithm to automatically find the best configuration. Do it for classes chosen in Task 3. Run your new system on a webcam and see if now it performs better than in Task 3.

Practical 4: Deliverables

- **40000-level students:** no need to submit your codes
- **60000-level students:** submit your code for Task 4 to Dropbox
- Discuss your observations in class, or send an e-mail to aczajka@nd.edu by **Tuesday, Oct. 20, 11:59pm** with a few sentences per task that summarize your observations