4-6 Wednesday – 210-GD3

# Special topics in Computer Science INT3121 20

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.

Email: ngocdiep@vnu.edu.vn

Slide & Code: https://github.com/chupibk/INT3121-20

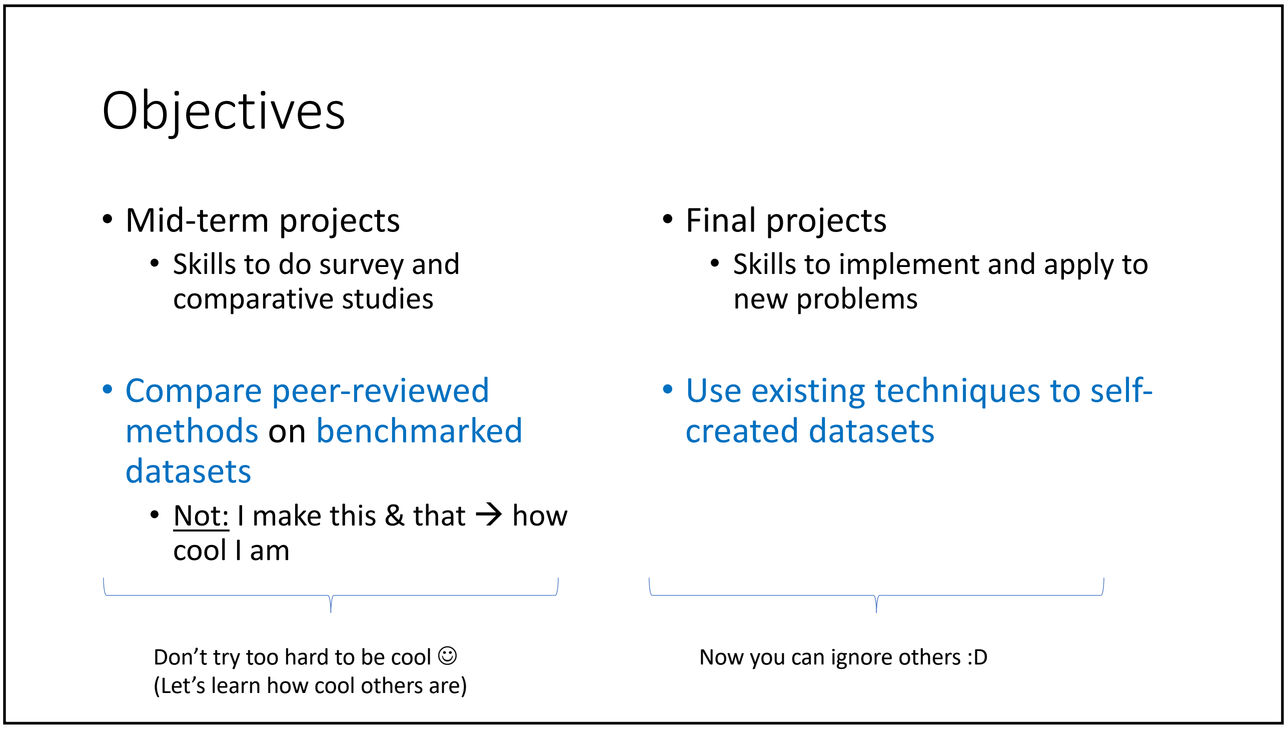
## Week 4 recall

- Generator: generate object data one at a time and only when asked
- Training with generator:
  - model.fit_generator()
  - Make sure the generator input return values whenever called
    - Data are flowed non-stop
- Augmentation:
  - Operators: blur, sharpen, edge detection, adding noise, invert, pad, crop, flip, resize...
- Generator + augmentation together:
  - keras.preprocessing.image.ImageDataGenerator()
    - .flow(x,y)
    - .flow_from_directory(directory)

## Registration (deadline: Oct 2, 2019 23:59)

- Link: https://forms.gle/BgAWdsCjHgD1nwoA6
- Group name
- Leader email
- Member IDs
- Chosen dataset
- Desired presentation date
- References: → submit before 1 week
  - Research papers (pdfs)
  - Links
  - Books (pdfs if possible)
- Presentation materials & code → submit before 1 day

## Mid-term != final projects



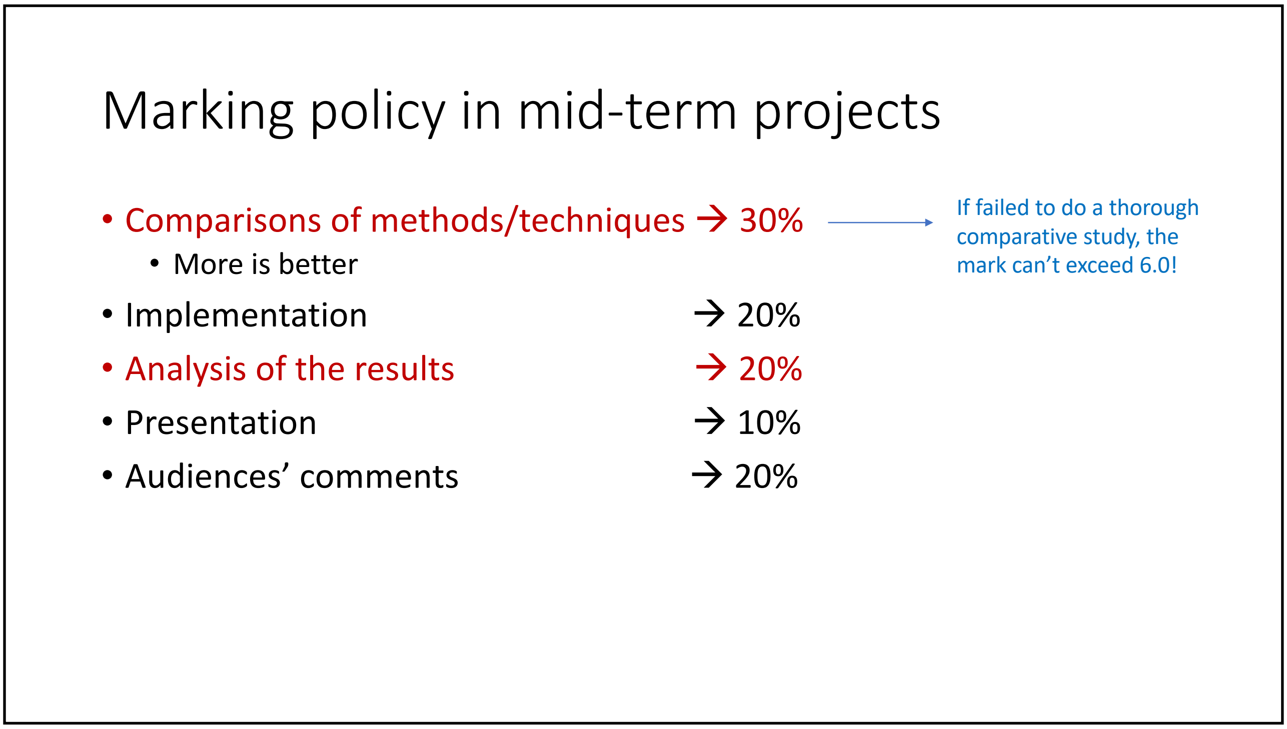
# Objectives

- Mid-term projects
  - Skills to do survey and comparative studies
- Compare peer-reviewed methods on benchmarked datasets
  - Not: I make this & that → how cool I am

Don't try too hard to be cool ☺
(Let's learn how cool others are)

- Final projects
  - Skills to implement and apply to new problems
- Use existing techniques to self-created datasets

Now you can ignore others :D



# Marking policy in mid-term projects

- Comparisons of methods/techniques → 30%
  - More is better
- Implementation → 20%
- Analysis of the results → 20%
- Presentation → 10%
- Audiences' comments → 20%

If failed to do a thorough comparative study, the mark can't exceed 6.0!

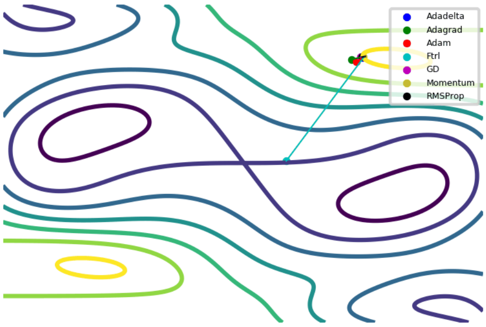
# Week 5: Transfer learning

## Image classification with convolutional neural networks

| Week | Content | Class hour | Self-study hour |
|---|---|---|---|
| 1<br>28/8/2019 | Introduction<br>Image classification problem and its applications<br>A toy problem with CIFAR10 | 2 | 1 |
| 2<br>(4/9/2019) | CNN model architectures and visualization | 2 | 1 |
| 3<br>(11/9/2019) | Training and tuning parameters<br>Automatic parameter learning | 2 | 1 |
| 4<br>(18/9/2019) | Data augmentation<br>Data generator | 2 | 2-6 |
| 5<br>(25/9/2019) | Transfer learning | 2 | 2-6 |
| 6<br>(2/10/2019) | Multi-output image classification | 2 | 2-6 |
| 7<br>(9/10/2019) | Building a training dataset<br>How to write a report | 1 | 2-6 |
| 8, 9, 10, 11 | Seminar: Bag of tricks with CNN<br>(as mid-term tests) | 1 | 2-6 |
| 12, 13, 14 | Final project presentations | 1-3 | 2-6 |
| 15 | Class summarization | 1 | open |

# Transfer learning

- Transfer learning = a technique where a model trained on one task is exploited in another task
- Weights of a Convolutional Network are often random initialized
  - Starting of optimizer to find global minimum for loss function
- Transfer learning -> a way to reuse weights



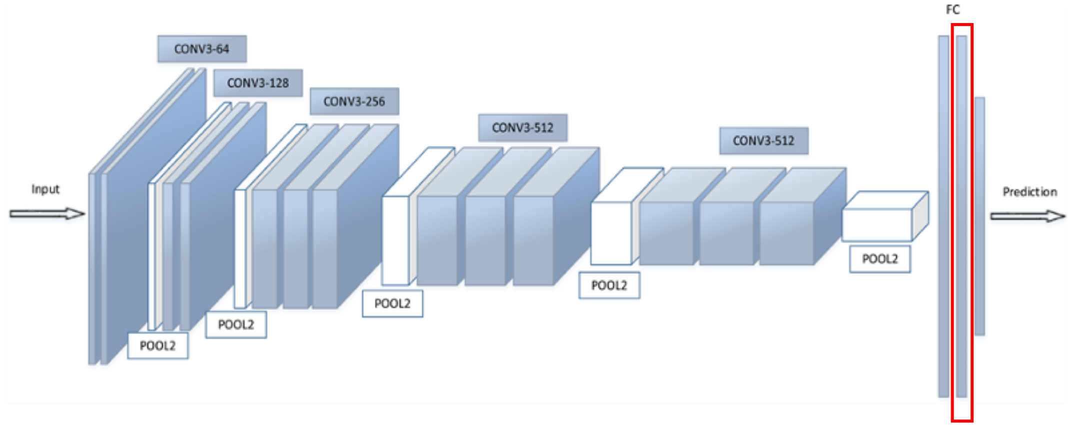
Visualization credit: https://github.com/Jaewan-Yun/optimizer-visualization

# Three major Transfer learning scenarios

1. ConvNet as fixed feature extractor
   - Remove (some) last fully-connected layer then run forward pass to extract features
2. Fine-tuning the ConvNet
   - Do backpropagation on some last layers only
3. Pretrained models
   - Do backpropagation on the whole model but with pretrained weights

Reference: http://cs231n.github.io/transfer-learning/

# ConvNet as fixed feature extractor



Take this out as features
(aka, CNN codes)

# What to do with features?

- Image classification using other machine learning techniques
  - E.g., SVM
- Image retrieval
  - Similarity-based
- Image clustering
  - Group images into clusters

# Assumption of Fine-tuning

- Earlier features of a ConvNet contain more generic features (e.g., edge, texture, color blob)
  - → should be useful to many task
- Later layers of the ConvNet are more specific to the task

# When & how to use which transfer learning?

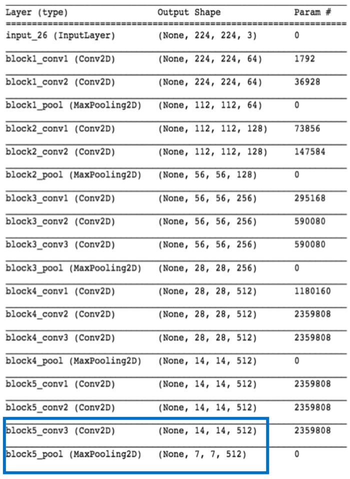
Size of new dataset

Type of new dataset

| | Small | Large |
|---|---|---|
| Similar | - Best: ConvNet as fixed feature extractor, then train with a linear classifier<br>- Fine-tune may cause overfitting | - fine-tune is okay |
| Very different | - Best: Extract earlier features in a ConvNet, then train with a linear classifier | - Initialize the network with weights from a pretrained model |

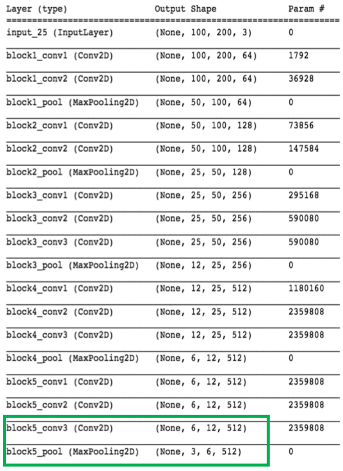
# Size of input image?

VGG16 with original
224x224 input



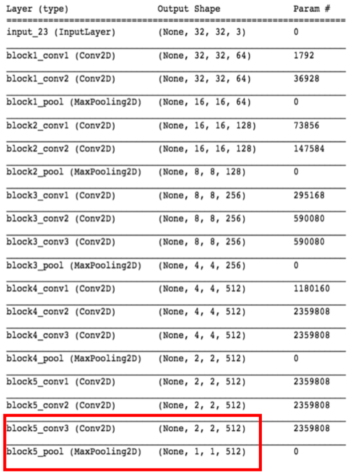
```
Layer (type)                 Output Shape              Param #
=================================================================
input_26 (InputLayer)        (None, 224, 224, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
```

100x200 input



```
Layer (type)                 Output Shape              Param #
=================================================================
input_25 (InputLayer)        (None, 100, 200, 3)       0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 100, 200, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 100, 200, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 50, 100, 64)       0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 50, 100, 128)      73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 50, 100, 128)      147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 25, 50, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 25, 50, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 25, 50, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 25, 50, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 12, 25, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 12, 25, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 12, 25, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 12, 25, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 6, 12, 512)        0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 6, 12, 512)        2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 6, 12, 512)        2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 6, 12, 512)        2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 3, 6, 512)         0
```

32x32 input



```
Layer (type)                 Output Shape              Param #
=================================================================
input_23 (InputLayer)        (None, 32, 32, 3)         0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 32, 32, 64)        1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 16, 16, 64)        0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 8, 8, 128)         0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 8, 8, 256)         295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 4, 4, 256)         0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 4, 4, 512)         1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 4, 4, 512)         2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 2, 2, 512)         0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 2, 2, 512)         2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 1, 1, 512)         0
```

If continue, run out of data