Special topics: Convolutional Neural Networks

Lecturer: Nguyen Thi Ngoc Diep, Ph.D.
Email: ngocdiep@vnu.edu.vn
Slide & Code: https://github.com/chupibk/INT3414_22

Schedule

Week	Content	Class hour	Self-study hour
1	Introduction CNNs in Computer Vision	2	1
2	~~Foundations of CNNs~~ ~~Case study: Image classification problem~~ Basics of Neural networks Training with backpropagation Implementation with PyTorch	2	2-6
3	~~Training and tuning parameters~~ ~~Data augmentation - Data generator~~ Foundations of CNNs Transfer learning Mid-term assignment	2	2-6
4	Object detection	2	2-6
5	Segmentation	2	2-6
6, 7	Mid-term presentations	2	2-6
8, 9	Advanced topics using CNNs	2	2-6
10, 11, 12	Final project presentations	1	2-6
13	Class summarization	1-3	open

Mid-term assignment

- Individual report:
  - Each student chooses a paper and write a report of 2-page A4
- Group presentation
  - Students choose a same paper will automatically form a group
  - Present your "discovery" at the class ☺
- Check this for more information:
  - https://forms.gle/ZUZgtHL6rLq7RNjU9

11 groups

https://www.dropbox.com/sh/hl6hkdpo2bjemyk/AADpE5u-2TicqdhgS9ajiLqya?dl=0

6
1. VGG
2. MobileNet
3. ResNet
4. ShuffleNet
5. GoogLeNet
6. Inception-v3
7. Inception-v4

Present together

Datasets: CIFAR10

7
8. Xception
9. ResNext
10. DenseNet
11. EfficientNet

https://forms.gle/ZUZgtHL6rLq7RNjU9

- What you need to do:
  1. Choose a paper (if the choices among students are too imbalanced, a paper will be assigned to you randomly)
  2. Read the paper carefully
  3. Implement the model and train it for CIFAR-10 dataset.
  4. Write a report about your discovery (using the below template)
  5. Submit the report (submission method will be announced later)
  6. Group with other students who read the same paper to make an awesome presentation

Midterm report template

1. Motivation
Describe the motivation of the paper

2. Method
Describe how authors design the architecture or algorithm to achieve that goal.
---> Demonstrate it by an interesting figure, diagram or code!

3. Evaluation
Describe the experiments the authors use to showcase the advantages of their proposed method.
And describe the disadvantages of the method (either stated by the authors or by your observation)

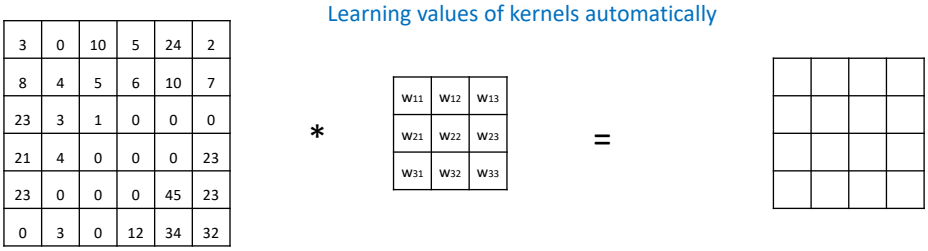
4. Reproduction
Describe your code and the results when you train the model for CIFAR10 classification dataset.
Note: it's okay to re-use code of others, but don't forget to make a reference and its correctness.

Marking policy for mid-term projects

- Individual report → 70%
  - Describe clearly the motivation, design idea → 40%
  - Able to reproduce the code → 30%
- Group presentation → 30%
  - Able to do teamwork → 10%
  - Clear presentation → 10%
  - Rank on CIFAR10 dataset (among all groups) → 10%

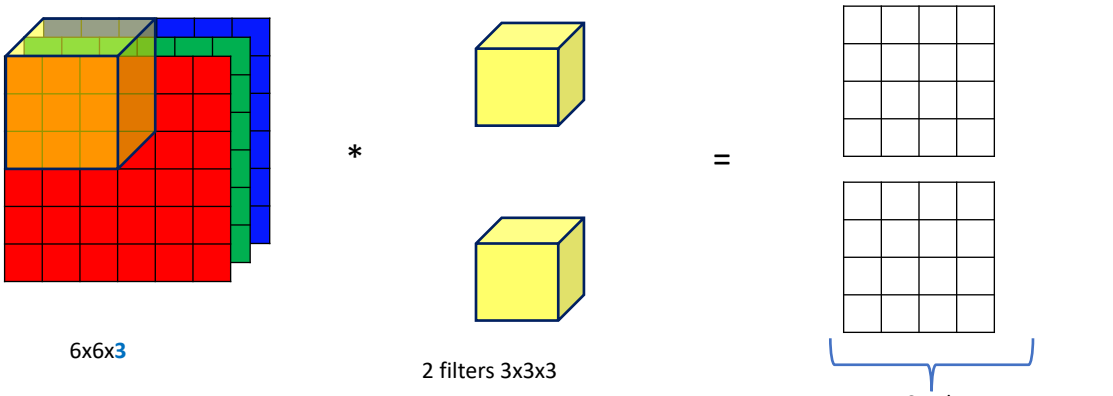
Recall week 3: CNNs foundations

Learning values of kernels automatically



- Hyperparameters:
  - Filter size
  - Pad size
  - Stride value
  - Number of filters

Recall week 3: Convolution over volume & multiple filters



Recall week 3: Parameters of a Convolution layer

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters: K,
  - Spatial extent (filter size): F,
  - Stride: S,
  - Amount of zero padding: P.
- Produces a volume of size $W_2 \times H_2 \times D_2$

$$W_2 = (W_1 - F + 2P)/S + 1$$
$$H_2 = (H_1 - F + 2P)/S + 1$$
$$D_2 = K$$

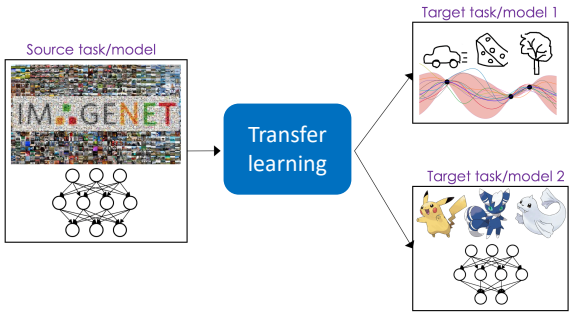
- Introduces $F \cdot F \cdot D_1$ weights per filter
- For a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases

Reference: http://cs231n.github.io/convolutional-networks/

Transfer learning

Motivation of 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



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/

An example of transfer learning

1. Rebuild weights from COCO
2. Freeze backbone, train neck and heads
3. Unfreeze 4th conv group
4. Unfreeze 3rd conv group
5. Full train except stem
6. Freeze backbone, train neck and heads on large batch (on V100 32Gb)

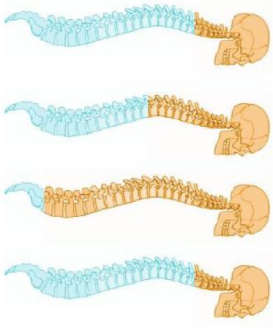
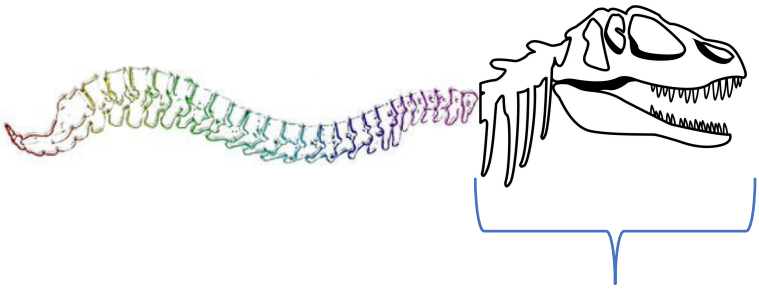


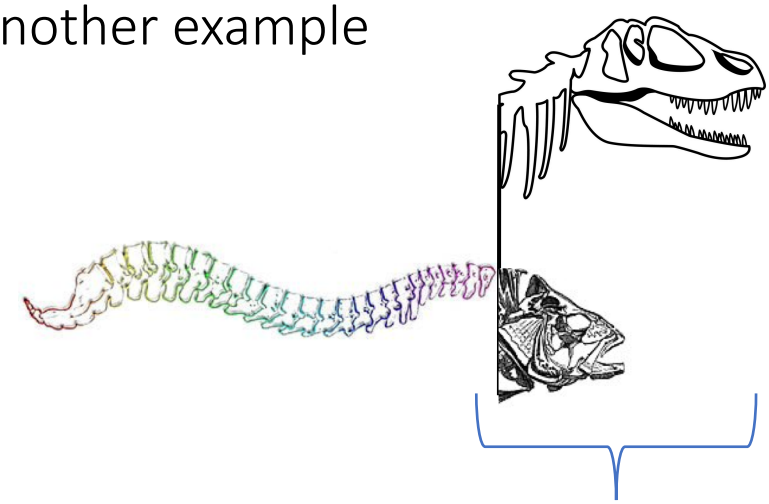
Illustration credit: ?

Another example



New head with some more “neck” layers

Another example

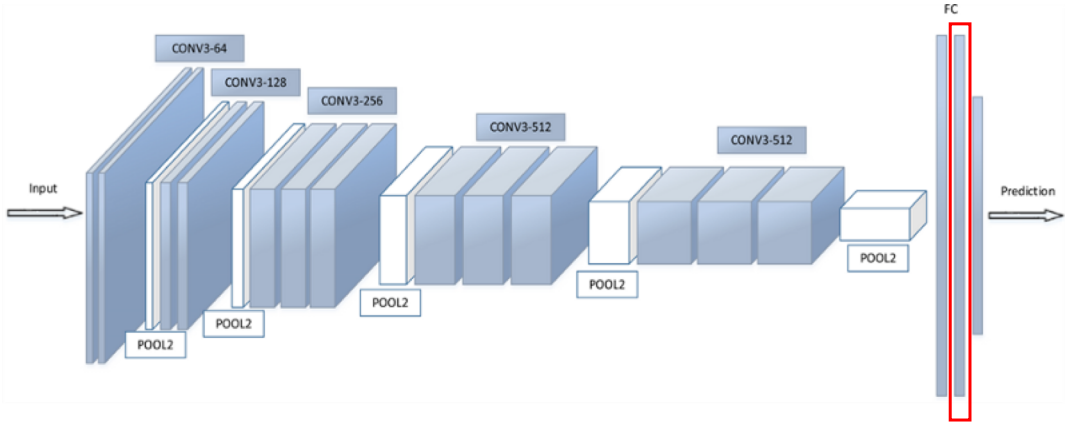


Multiple heads

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

ConvNet as fixed feature extractor



Take this out as features (aka, CNN codes)

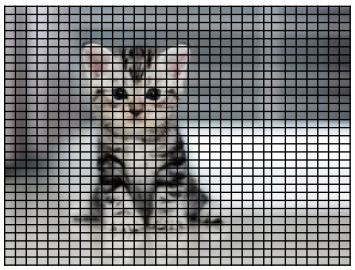
What to do with features?

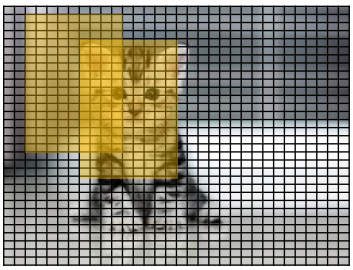
- Image classification using other machine learning techniques
  - E.g., SVM
- Image retrieval
  - Similarity-based
- Image clustering
  - Group images into clusters

CNNs for Object detection

Vanilla object detection

- Using sliding window and treat it like classification

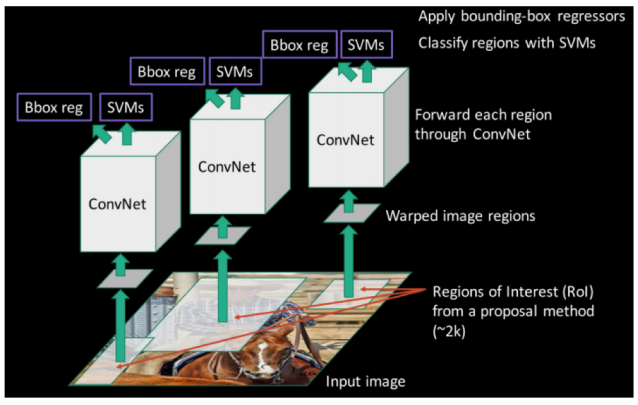




Limitation:
- Size of window
- Large number of “candidate” regions
- Unbalanced between “True”, “False” classes

R-CNN

Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014.



Binary SVM classifier for each class

Limitation:
- Fixed selective search algorithm for region proposals
- Number of regions is still large (2k)
- Large inference time (~47s)

https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/object_localization_and_detection.html

Bounding box regression (Class-specific)

P: Proposal box (center coordinates, width, height) $P = (P_x, P_y, P_w, P_h)$

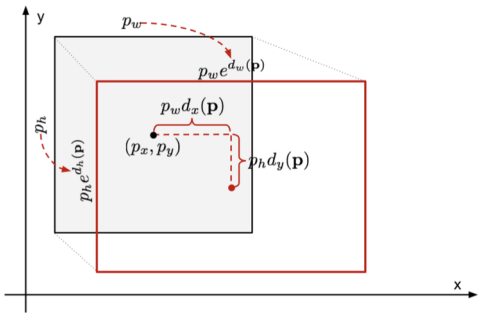
G: Ground truth box

$\hat{G}$: Predicted bounding box

Four learnable functions: $d_*(P)$

Transform P into to $\hat{G}$:

$$\hat{G}_x = P_w d_x(P) + P_x$$
$$\hat{G}_y = P_h d_y(P) + P_y$$
$$\hat{G}_w = P_w \exp(d_w(P))$$
$$\hat{G}_h = P_h \exp(d_h(P))$$

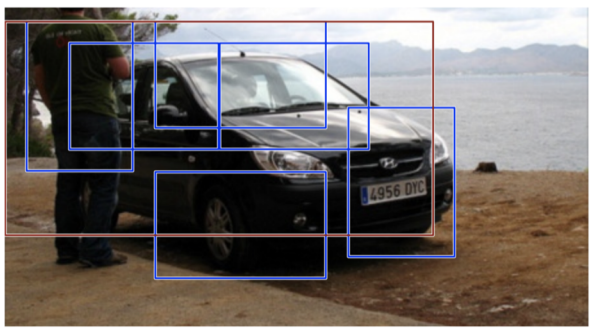


Solve by minimizing the SSE loss (with regularization) between predicted and ground truth boxes

Non-maximum suppression

Model will find multiple bounding boxes for the same object

- Sort all the bounding boxes of the same object category by confidence score;
- Discard boxes with low confidence scores
- *While* there is any remaining bounding box, repeat the following:
  Greedily select the one with the highest score.
  Skip the remaining boxes with high IoU (i.e. > 0.5) with previously selected one.



Before non-max suppression

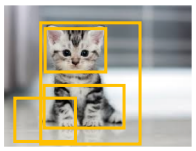
After non-max suppression

Source: Deformable Part Model,

Fast R-CNN

Girshick, Ross. "Fast r-cnn." *Proceedings of the IEEE international conference on computer vision*. 2015.

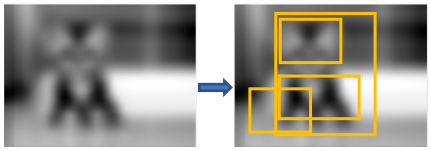
R-CNN



Each proposal is fed to the CNN to extract feature

Faster R-CNN

CNN to extract a feature map

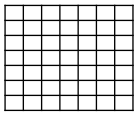


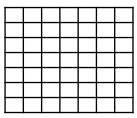
ROI pooling for each proposal

Fast R-CNN: ROI pooling

Max pooling

Output size: 7x7 (fixed)





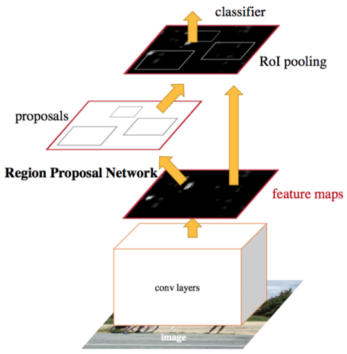
....

Feed to classifier and regressors

Faster R-CNN

Ren, Shaoqing, et al. "Faster r-cnn: Towards real-time object detection with region proposal networks." *Advances in neural information processing systems*. 2015.

Motivation: Learn to get region proposals



Fast R-CNN: Classify each proposal and refine proposal location

RPN: Gives a set of rectangular object proposals, each with an objectness score

Deal with scales

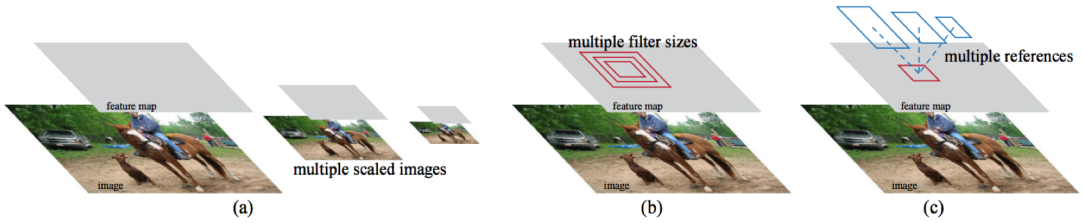


Figure 1: Different schemes for addressing multiple scales and sizes. (a) Pyramids of images and feature maps are built, and the classifier is run at all scales. (b) Pyramids of filters with multiple scales/sizes are run on the feature map. (c) We use pyramids of reference boxes in the regression functions.

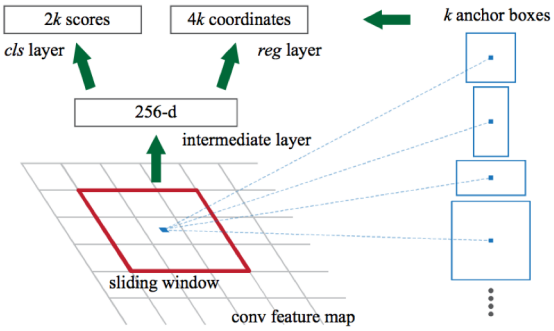
Source: Faster R-CNN

Anchors

Anchors are fixed bounding boxes that are placed throughout the image with different sizes and ratios that are going to be used for references

Pre-define:
- a set of sizes (e.g., 64px, 128px, 256px)
- a set of ratios between width and height of boxes (e.g., 0.5, 1, 1.5)

A reference box: x_{center}, y_{center}, *width*, *height*
Learn to predict: $\Delta_{x_{center}}$, $\Delta_{y_{center}}$, Δ*width*, Δ*height*

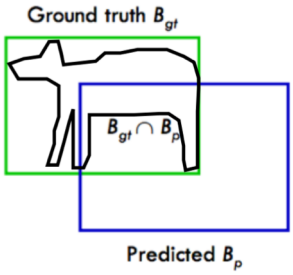


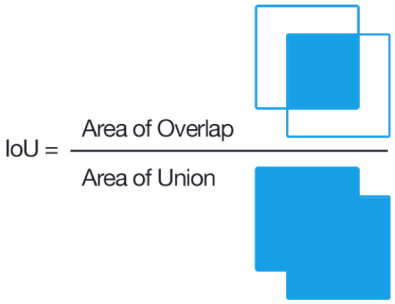
Other object detection models

- Many many...
  - Pooling Pyramid Network
  - YOLO v1/9000,v3 (You Only Look Once)
  - SNIPER (Efficient Multi-scale training)
  - SSD (Single Shot Multibox Detector)
  - FoveaBox
  - ... → https://paperswithcode.com/task/object-detection
- Similar task: Scene text detection

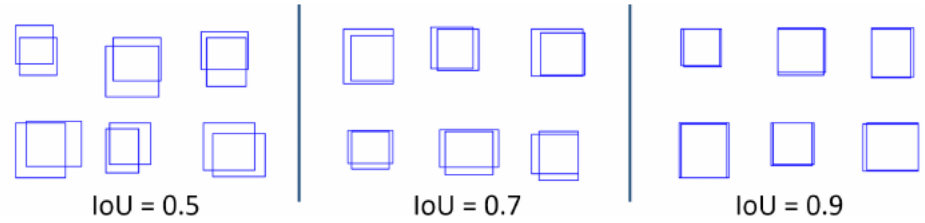
Evaluation metrics

Intersection over Union (IoU)





Bounding box prediction accuracy



Evaluation metrics for binary classification

total N=150	Predicted YES	Predicted NO
Actual: YES	True positive TP = 80	False Negative FN = 25
Actual: NO	False Positive FP = 10	True negative TN = 35

$$precision = \frac{number\ of\ correct\ positive\ predictions}{number\ of\ positive\ predictions\ made}$$

$$= \frac{TP}{TP + FP} = \frac{80}{80 + 10} = 0.89$$

$$accuracy = \frac{number\ of\ correct\ predictions}{total\ number\ of\ predictions\ made}$$

$$= \frac{TP + TN}{N} = \frac{80 + 35}{150} = 0.7$$

$$recall = \frac{number\ of\ correct\ positive\ predictions}{number\ of\ all\ positive\ samples}$$

$$= \frac{TP}{TP + FN} = \frac{80}{80 + 25} = 0.76$$

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

Harmonic mean between precision and recall

Evaluation metrics for multi-class classification

	Predicted YES	Predicted NO
Actual: YES	True positive	False Negative
Actual: NO	False Positive	True negative

Recall = TP/Total actual positive

Precision = TP/Total predicted positive

	GoldLabel_A	GoldLabel_B	GoldLabel_C	
Predicted_A	30	20	10	TotalPredicted_A=60
Predicted_B	50	60	10	TotalPredicted_B=120
Predicted_C	20	20	80	TotalPredicted_C=120
	TotalGoldLabel_A=100	TotalGoldLabel_B=100	TotalGoldLabel_C=100	

Recall label X = TP_X / Total actual X

Precision label X = TP_X / Total predicted_X

Ref: http://text-analytics101.rxnlp.com/2014/10/computing-precision-and-recall-for.html

sklearn.metrics

sklearn.metrics.precision_score(*y_true*, *y_pred*, *labels=None*, *pos_label=1*, *average='binary'*, *sample_weight=None*)

average : *string, [None, 'binary' (default), 'micro', 'macro', 'samples', 'weighted']*

This parameter is required for multiclass/multilabel targets. If `None`, the scores for each class are returned. Otherwise, this determines the type of averaging performed on the data:

`'binary'`:
Only report results for the class specified by `pos_label`. This is applicable only if targets (`y_{true,pred}`) are binary.

`'micro'`:
Calculate metrics globally by counting the total true positives, false negatives and false positives.

`'macro'`:
Calculate metrics for each label, and find their unweighted mean. This does not take label imbalance into account.

`'weighted'`:
Calculate metrics for each label, and find their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

`'samples'`:
Calculate metrics for each instance, and find their average (only meaningful for multilabel classification where this differs from `accuracy_score`).