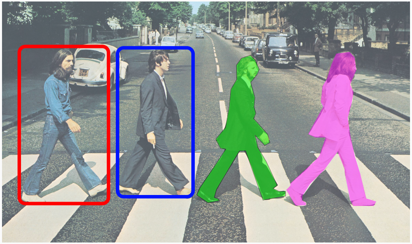
ECE 285

Machine Learning for Image Processing

Chapter V – Detection, Segmentation, Captioning

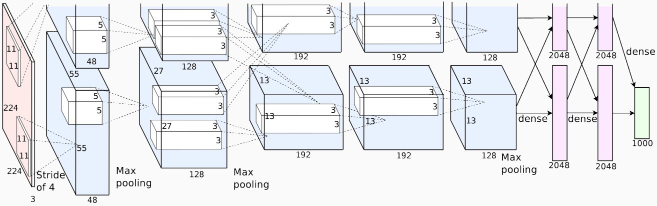
Charles Deledalle

November 16, 2019



4 lads from Liverpool

Recap of CNNs for classification



- CNNs are ANNs using convolutions instead of full matrix-vector products,
- Use pooling layers between layers to increase their effective receptive fields,
- Successively reduce spatial dimensions until the tensor is (almost) *flat*,
- A classifier (generally 3 layers ANNs) is finally plugged after this,
- Various architectures: Inception module, ResNet, DenseNet, ...

Detection, Segmentation, Captioning

Principle tasks

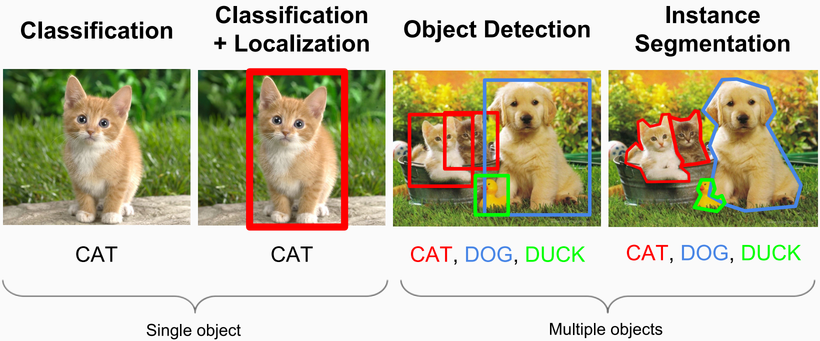
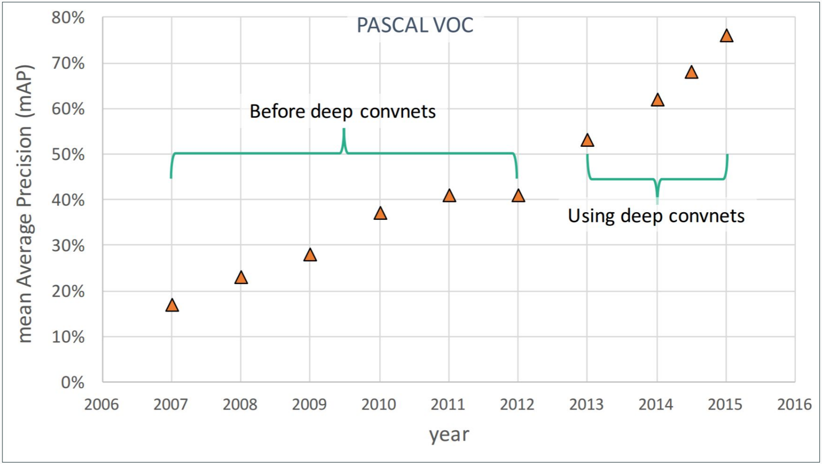


Image captioning: `'Animals posing for picture.'`

Influence of Deep Learning



Multi-object detection challenge

Classification + localization

Classification + localization

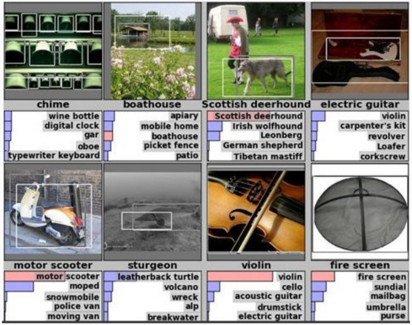
Classification + localization

Localization: find the bounding box (bbox) around the (single) object.

Regression problem: predict 4 values encoding the bbox size and location, usually: left (x), top (y), width (w), height (h).

Classification + localization: ImageNet

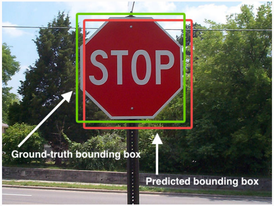
- 1,000 classes (same as classification),
- Each image has one bounding box,
- About 800 images per class,
- Algorithm produces 5 classes,
- Evaluation metric: at least **one correct class prediction** and one bbox with **at least .5 Intersection-over-Union (IoU)**.



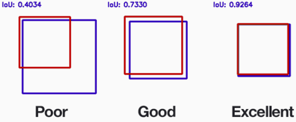
(Source: Ric Poirson)

Intersection-over-Union (IoU)

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}$$



$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



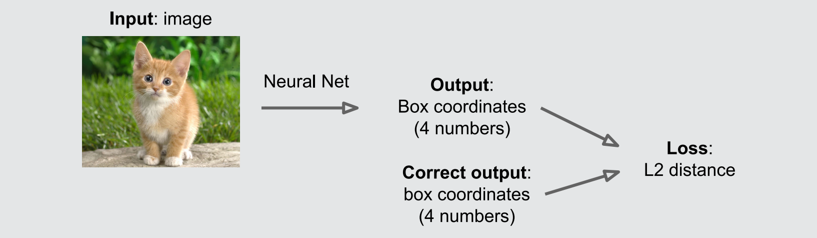
Naive approach

- Choose a classifier (AlexNet, VGG, GoogLeNet, ...),
- Extract all possible bounding boxes,
- Rescale their contents to the size of the network image input,
- Classify each of them,
- Select the one with the maximum confidence level.

Probably works great, but impractical. It would be too slow:
$\rightarrow$ Need to test many positions and scales,
and use a computationally demanding classifier (CNN).

Localization as regression

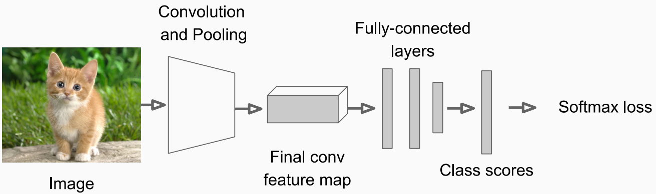
- **Classification**
  - Input: image
  - Output: class labels
  - Loss: cross-entropy
- **Localization**
  - Input: image
  - Output: bounding box (x, y, w, h)
  - Loss: MSE / ℓ_2^2 error



- **Classification+Localization:** do both

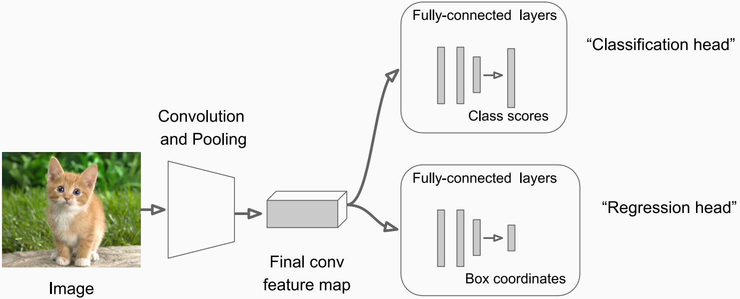
(Source: Ric Poirson)

Simple recipe for Classification + Localization



- Step 1: Train/download a classification model (AlexNet, ResNet, . . .),

Simple recipe for Classification + Localization



- Step 1: Train/download a classification model (AlexNet, ResNet, ...),
- Step 2: Attach a new fully connected regression head,
- Step 3: Train the regression head only with SGD and ℓ_2^2 loss,
- Step 4: At test time, use both heads.

What to learn exactly?

- **Classification head:**

  1-of-K code with confidence levels $(0, 1)$

- **Regression head:**
  - Class agnostic:

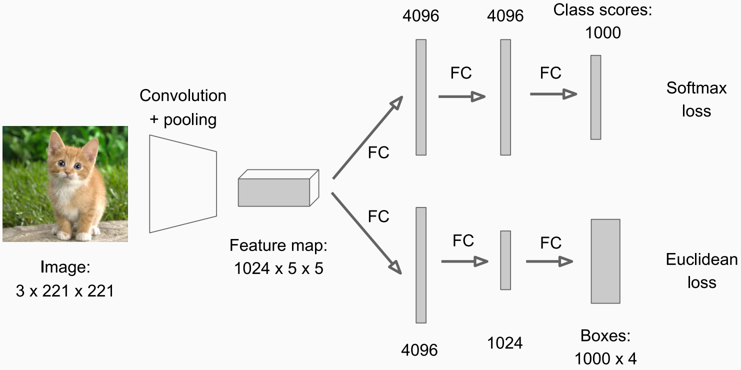
    4 numbers (one bounding box)

  - Class specific:

    $K \times 4$ (one box per class)

Being agnostic doesn't work as well:
the strategy to find the bounding-box must depend on the class.

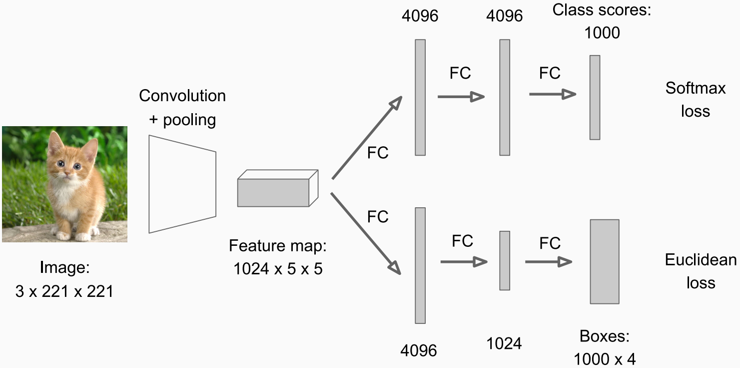
Overfeat (Sermanet *et al.*, 2013)



Tweak only the weights of the bbox branch:

$$E(\boldsymbol{W}_{\text{bbox}}) = \sum_{(\boldsymbol{x},d,x,y,w,h)\in\mathcal{T}} (x - \hat{x}_d)^2 + (y - \hat{y}_d)^2 + (w - \hat{w}_d)^2 + (h - \hat{h}_d)^2$$

Overfeat (Sermanet *et al.*, 2013)



The softmax layer provides the confidence level of each bounding box.

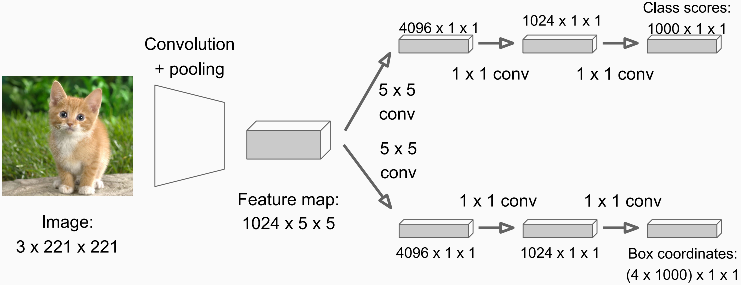
Training: augment the data by random shifts and multi-scales, but keep only the ones with 50% overlap with the desired bounding-box.

Testing: use multi-scale sliding windows and a greedy merge strategy.

Overfeat – Sliding windows

How to get classification results for all sliding windows without running the classifier multiple times?

1. Convert FC layers to 1×1 convolutions and fine tune



The size of the sliding window will be the network input size.

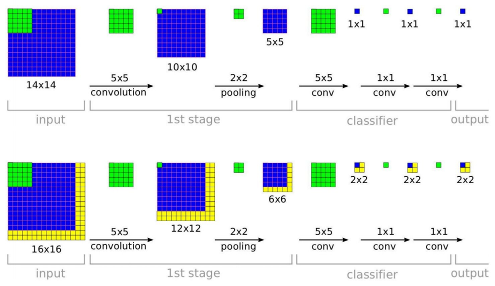
This is called a Fully Convolutional Network (FCN).

Overfeat – Sliding windows

2. Apply the network on the larger image

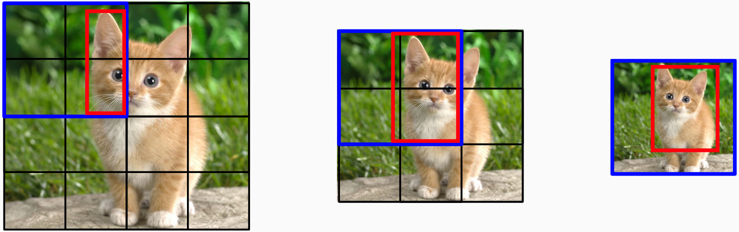
Training time: Small image, 1 x 1 classifier output

Test time: Larger image, 2 x 2 classifier output, only extra compute at yellow regions



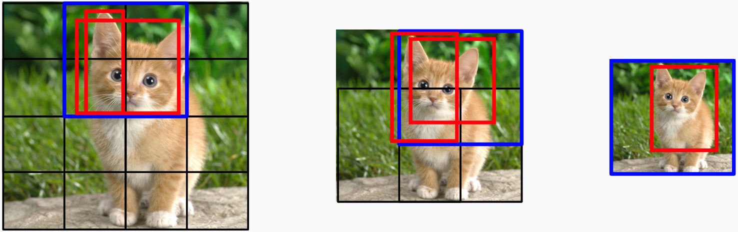
As all layers are convolutions and pooling, the network parameters do not depend anymore on the input image size. If a larger image is given, multiple localized answers will be produced with a smaller computational overhead (the stride at which the window slides depends on the pooling layers).

Overfeat – Greedy merge strategy



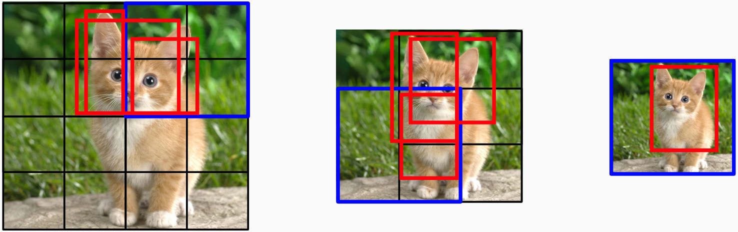
1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,

Overfeat – Greedy merge strategy



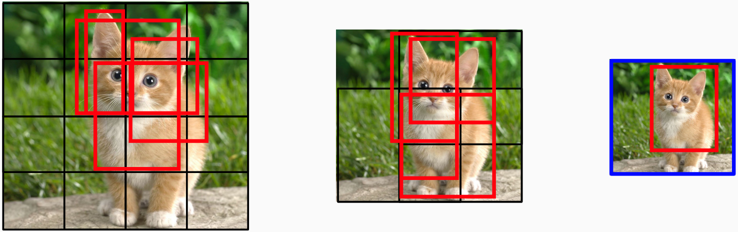
1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,

Overfeat – Greedy merge strategy



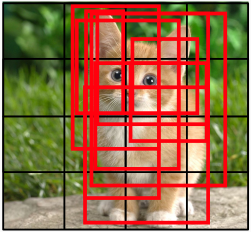
1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,

Overfeat – Greedy merge strategy



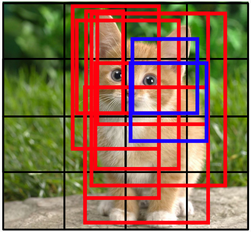
1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,

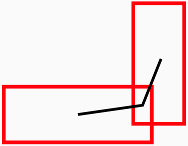
Overfeat – Greedy merge strategy



1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,

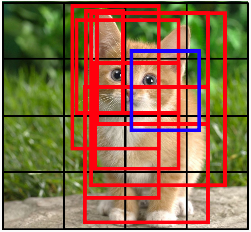
Overfeat – Greedy merge strategy

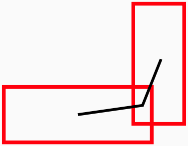




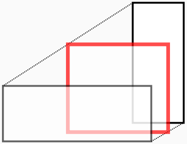
1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,
2. Look for the two closest bounding boxes: distances to their intersection.

Overfeat – Greedy merge strategy





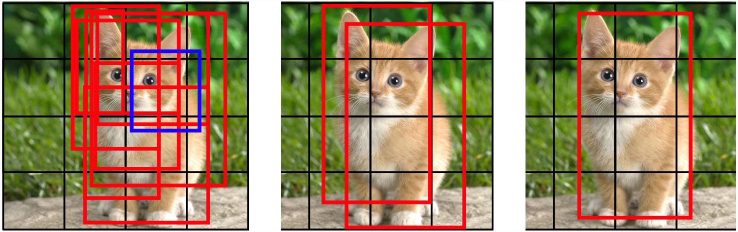
Measure of distance



Strategy for merging

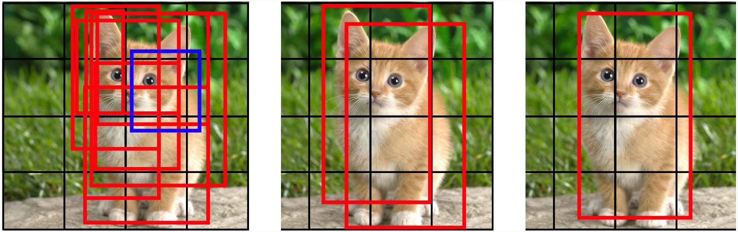
1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,
2. Look for the two closest bounding boxes: distances to their intersection.
3. Merge them by averaging their coordinates and scores,
4. Update the set of candidates by replacing them by their merged version,

Overfeat – Greedy merge strategy



1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,
2. Look for the two closest bounding boxes: distances to their intersection.
3. Merge them by averaging their coordinates and scores,
4. Update the set of candidates by replacing them by their merged version,
5. Go back to Step 2 until one bounding box remains.

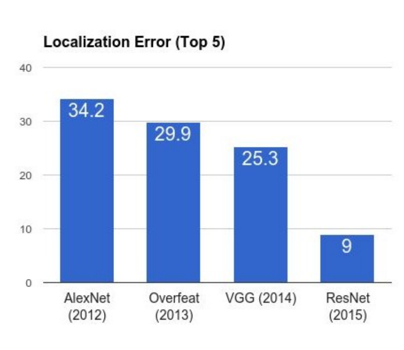
Overfeat – Greedy merge strategy



1. Use the network to extract a set of candidate bounding boxes for each scale and sliding window,
2. Look for the two closest bounding boxes: distances to their intersection.
3. Merge them by averaging their coordinates and scores,
4. Update the set of candidates by replacing them by their merged version,
5. Go back to Step 2 until one bounding box remains.

→ **Winner ILSVRC LOC 2013!**

Classification + localization – ImageNet results



AlexNet: Localization method not published

Overfeat: Multiscale convolutional regression with box merging

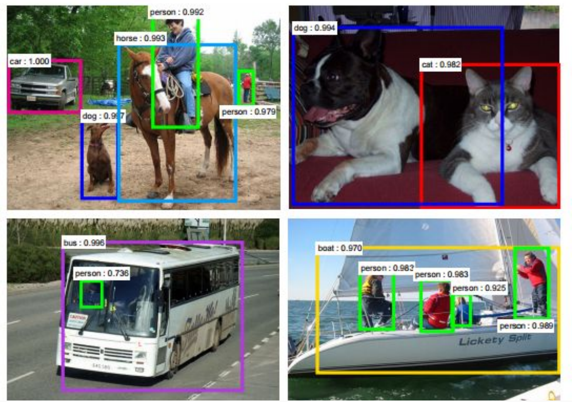
VGG: Same as Overfeat, but fewer scales and locations; simpler method, gains all due to deeper features

ResNet: Different localization method (RPN) and much deeper features

Most recent techniques use a classifier in predefined windows.
These techniques aim directly at solving
the multi-object detection task.

(Multi-)Object detection

Object detection



- **Goal:** detect and localize all objects of the scene,
- **Problem:** need to test many positions and scales,
- **Solution:** only look at a tiny subset of possible positions.

Object detection – Datasets and challenges

PASCAL Visual Object Classification (VOC):

- Since 2005
- $\sim$10,000 images, 20 categories
- Evaluation: mAP with IoU $\geqslant .5$

ILSVRC Detection

- Since 2013
- $\sim$500,000 images, 200 categories
- Evaluation: mAP with IoU $\geqslant .5$

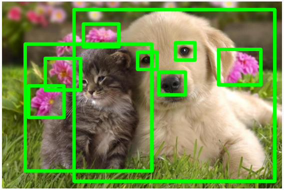
Microsoft Common Objects in COntext (COCO)

- Since 2015
- $\sim$200,000 images, 80 object categories
- Evaluation: mAP averaged over IoU $\geqslant .50 : .05 : .95$

PASCAL VOC'2010

Basic region proposals

- Pre-select image regions that are likely to contain objects,
- Use a class agnostic object detector for this task,

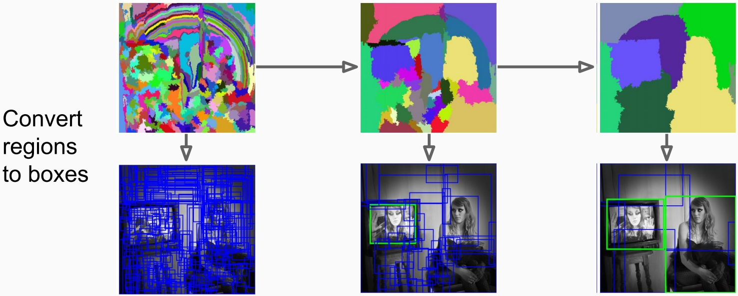


- Pick a fast one, not necessarily based on machine learning,
- Prefer having too many regions (FPs) rather than missing some (FNs).

Region proposals – Selective Search (Uijlings *et al.*, 2013)

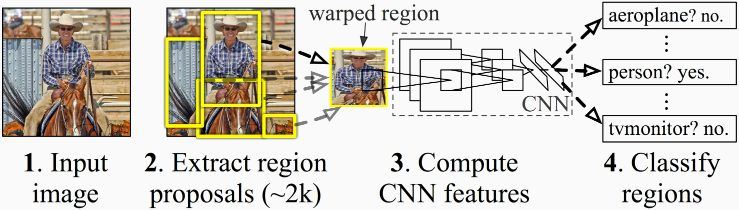
- Over-segmentation: decompose the image into small coherent regions,
  ex: based on colors (RGB, HSV, YCbCr, ...), textures, super-pixels.

- Bottom-up segmentation: merge similar regions at multiple scales.



- Rescale/warp and provide all such regions as inputs of a classifier.

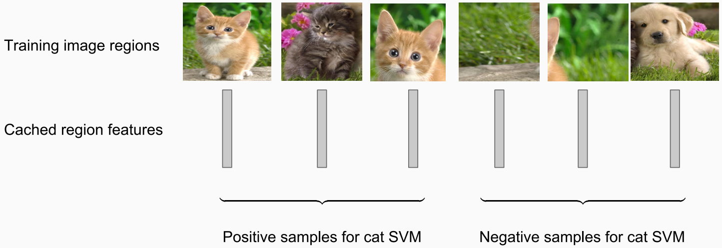
R-CNN (Girschick *et al*, 2014)



- Use Selective Search to extract 2,000 regions and warp them,
- Use AlexNet or VGG-16 for feature extraction,
- Replace the last FC layers (classifier) by one binary SVM per class, why?
  - ILSVRC classification has 1000 classes, but detection challenges have less: 20 for PASCAL VOC, 200 for ILSVRC LOC, 80 for MS COCO.
  - Classification challenges do not have a class 'background' (no relevant objects) since all images contain an object, but sub-regions don't.

R-CNN – Training

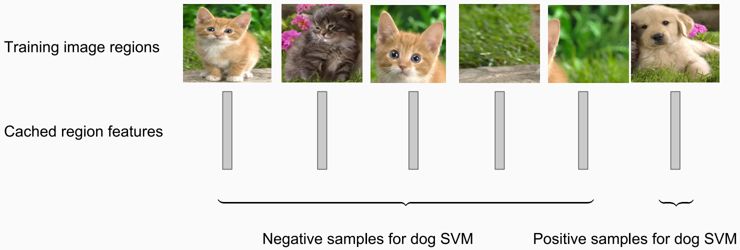
- Extract subregions for all training images and warp them,
- Run the CNN and save their features (tensors).



- Next, train one binary SVM per class to classify each feature tensor,
- Consider a region as positive if it overlaps the desired true bounding box with an IoU greater than .5, as background otherwise.

R-CNN – Training

- Extract subregions for all training images and warp them,
- Run the CNN and save their features (tensors).



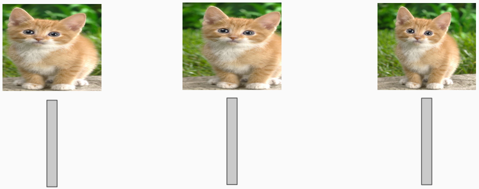
- Next, train one binary SVM per class to classify each feature tensor,
- Consider a region as positive if it overlaps the desired true bounding box with an IoU greater than .5, as background otherwise.

R-CNN – Training

As for Overfeat, learn also a regressor for refining the bounding box to make up for slightly wrong proposals.

Training image regions

Cached region features



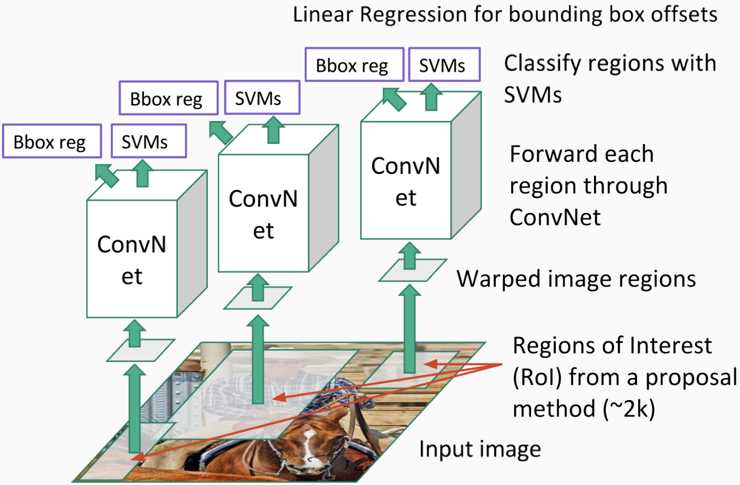
Regression targets
(dx, dy, dw, dh)
Normalized coordinates

(0, 0, 0, 0)
Proposal is good

(.25, 0, 0, 0)
Proposal too far to left

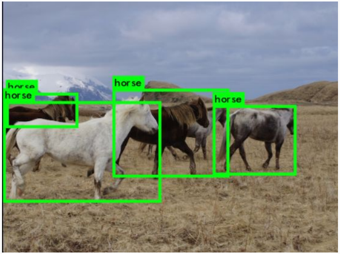
(0, 0, -0.125, 0)
Proposal too wide

R-CNN – Global architecture

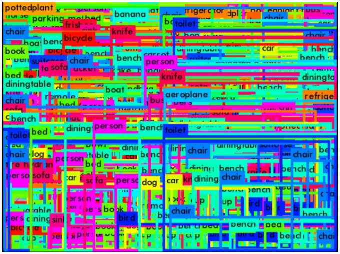


R-CNN – Non-Maximum Suppression (NMS)

We expect:

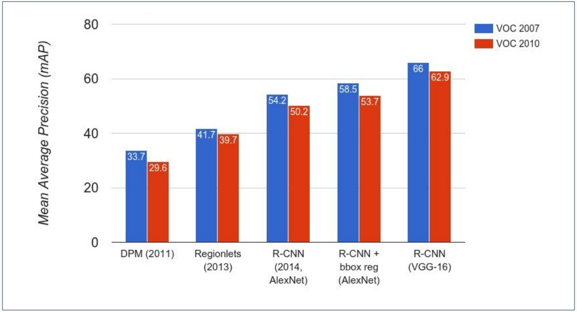


We get:



- **Score thresholding:** keep only bounding boxes with large confidence.
- **(Greedy) Non-maximum suppression:**
  1. Select the best scoring window,
  2. Remove windows too close to the selected one,
  3. Select the next best scoring window among remaining ones,
  4. Repeat to Step 2 until no more windows are removed.
- Many variants exist.

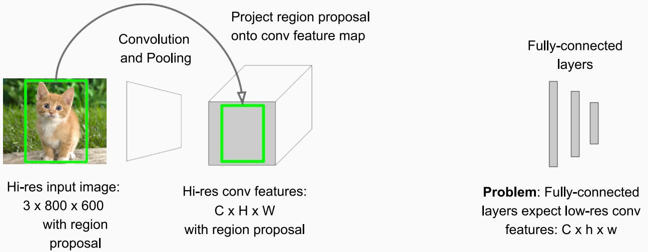
R-CNN – Results and issues



- Slow at test-time: need to run the CNN for each proposed region,
- CNN features may not be adapted in response to SVMs and regressors,
- Complex multistage training pipeline: SVM, bounding box regressor...
- Require a large disk storage at training (need to save feature tensors).

Fast R-CNN (Girshick, 2015)

In R-CNN, the same features are computed multiple times.
(since convolutions are translation invariant)



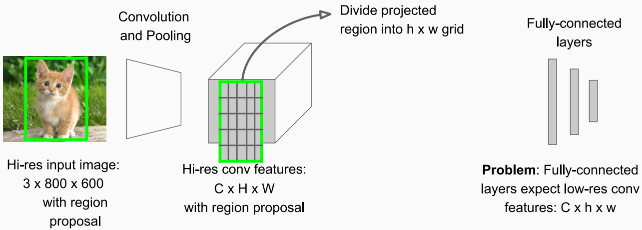
Region of Interest (RoI) pooling layer:

- Project each proposed region into the feature map,
- Divide each region into $h \times w$ grid (cells depend on region size),
- Perform max-pooling in each block to get a fixed size $h \times w$ feature.

CNN needs to be run only once instead of $2,000$ times!

Fast R-CNN (Girshick, 2015)

In R-CNN, the same features are computed multiple times.
(since convolutions are translation invariant)



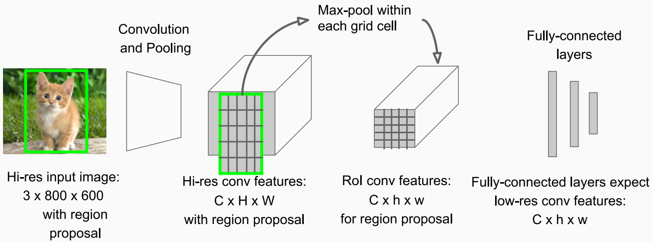
Region of Interest (RoI) pooling layer:

- Project each proposed region into the feature map,
- Divide each region into $h \times w$ grid (cells depend on region size),
- Perform max-pooling in each block to get a fixed size $h \times w$ feature.

CNN needs to be run only once instead of $2,000$ times!

Fast R-CNN (Girshick, 2015)

In R-CNN, the same features are computed multiple times.
(since convolutions are translation invariant)

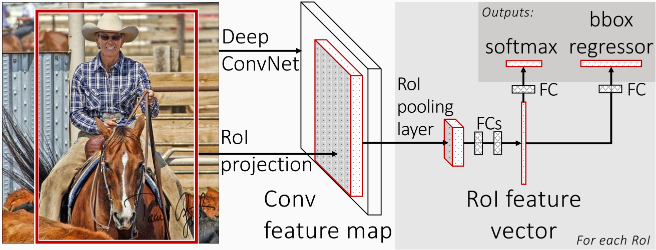


Region of Interest (RoI) pooling layer:

- Project each proposed region into the feature map,
- Divide each region into $h \times w$ grid (cells depend on region size),
- Perform max-pooling in each block to get a fixed size $h \times w$ feature.

CNN needs to be run only once instead of $2,000$ times!

Fast R-CNN (Girshick, 2015)



End-to-end training:

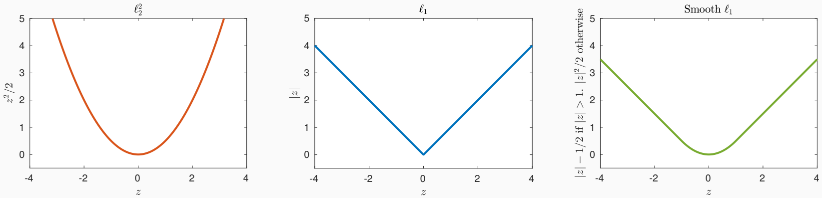
- Plug FC layers after the RoI pooling layer and fine tune,
- Incorporate both: classifier and bounding box regressor:
  - softmax: K Classes + 'background' class,
  - bbox regression: $K \times 4$ (class specific).

Fast R-CNN (Girshick, 2015)

Multitask loss: cross-entropy + bbox refinement

$$E = \underbrace{-p_d \log \hat{p}_d}_{\text{cross-entropy}} + \lambda \mathbf{1}_{d \neq 0} \underbrace{\left[|x - \hat{x}_d| + |y - \hat{y}_d| + |w - \hat{w}_d| + |h - \hat{h}_d| \right]}_{\ell_1 \text{ loss}}$$

- d: desired class label (background = 0),
- p_d / $\hat{p}_d$: one-hot codes and predicted probabilities for class d,
- x, y, w, h: desired bounding box,
- $\hat{x}_d, \hat{y}_d, \hat{w}_d, \hat{h}_d$: predicted bounding box for class d,
- $\lambda > 0$: hyperparameter balancing the two task losses.



Smooth ℓ_1: more robust than ℓ_2^2 (SSD) easier to optimize than ℓ_1.

Fast R-CNN – Results

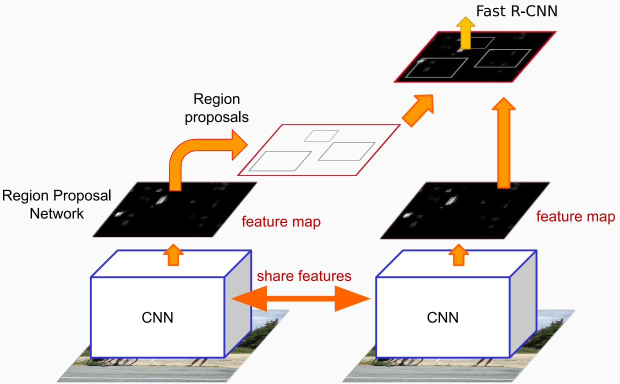
	R-CNN	Fast R-CNN	
mAP	66.0	66.9	better ☺
Training time	84 hours	9.5 hours	faster ☺
Test time per image	47 sec	0.32 sec	a lot faster ☺
+ Selective search	50 sec	2 sec	bottleneck ☹

(On PASCAL VOC 2007 and using VGG-16)

- Training is $8\times$ faster,
- Testing is $146\times$ faster but this does not include Selective Search,
- Only $25\times$ faster when including Selective Search.

Can we make the CNN do region proposals too?

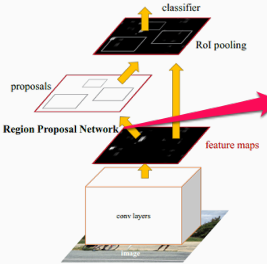
Faster R-CNN (Ren *et al.*, 2015)

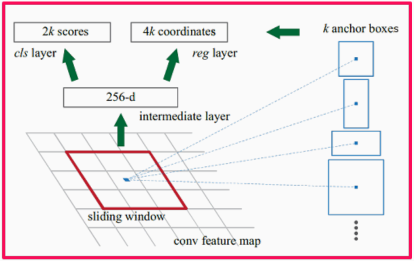


- Insert a Region Proposal Network (RPN) after the last convolutional layer,
- RPN is small and trained to produce region proposals directly,
- Next, use RoI pooling, classifier and bbox regressor (just like Fast R-CNN).

Faster R-CNN – RPN

- Slide a window on the feature map: $\begin{cases} \bullet \text{ classify into object/background,} \\ \bullet \text{ regress bbox locations.} \end{cases}$
- Position of the sliding window = rough localization,
- Box regression = refined localization.





- Use a predefined set of nine sliding windows called anchor boxes,
- Regression gives offsets from anchor boxes to proposed RoI,
- Classification gives the probability that each proposed RoI shows an object.

Faster R-CNN – Training & Results

Train everything together with four loss:

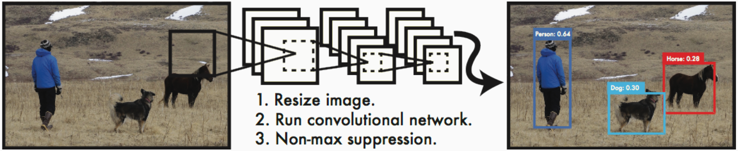
- RPN classification (keep anchor or not),
- RPN regression (anchor $\rightarrow$ proposed RoI),
- Fast R-CNN classification (over classes),
- Fast R-CNN regression (RoI $\rightarrow$ bounding box).

Results

	R-CNN	Fast R-CNN	Faster R-CNN
mAP	66.0	66.9	66.9
Test time per image with Selective search	50 sec	2 sec	0.2 sec

(On PASCAL VOC 2007 and using VGG-16)

YOLO: You Only Look Once (Redmon *et al.*, 2016)



- Proposal-free object detection pipeline,
- Learn directly a CNN predicting the bounding boxes:

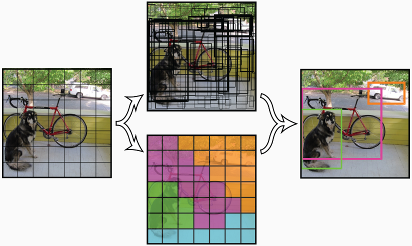
  448×448 Image $\rightarrow$ Bounding box coordinates and class probabilities,
- As the number of objects is unknown, decompose the image on a 7×7 grid, and predict 2 bboxes per cell (up to 98 bounding boxes),
- Perform non-max suppression.

Use features from the entire image to predict simultaneously each bounding box.

YOLO: You Only Look Once (Redmon *et al.*, 2016)

- Learn 2 bboxes per cell of a 7×7 grid:

  If the center of an object falls into a grid cell, that cell is responsible for detecting that object.



- Each cell contains:
  - Class probabilities

    $$p_k \quad \text{st} \quad p_k = \mathrm{Pr}(\mathrm{Class}_k | \mathrm{Object})$$

  - Two bounding boxes with confidences:

    $$2 \times (x, y, w, h, c) \quad \text{where} \quad c = \mathrm{Pr}(\mathrm{Object}) \times \mathrm{IoU}$$

    (up to two objects can have their center in the same cell)

- At test time, individual box confidence prediction

$$p_k c = \mathrm{Pr}(\mathrm{Class}_k | \mathrm{Object}) \times \mathrm{Pr}(\mathrm{Object}) \times \mathrm{IoU} = \mathrm{Pr}(\mathrm{Object}\ \&\ \mathrm{Class}_k) \times \mathrm{IoU}$$

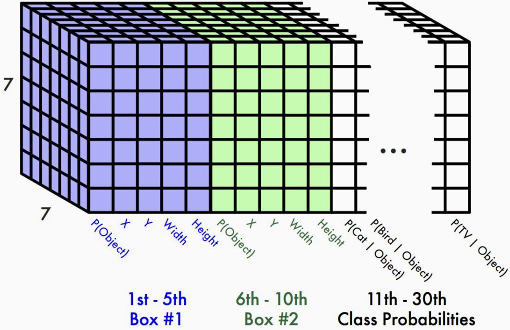
YOLO: You Only Look Once (Redmon *et al.*, 2016)

Each cell predicts:

- For each bounding box:
  - 4 coordinates (x, y, w, h)
  - 1 confidence value
- Some number of class probabilities

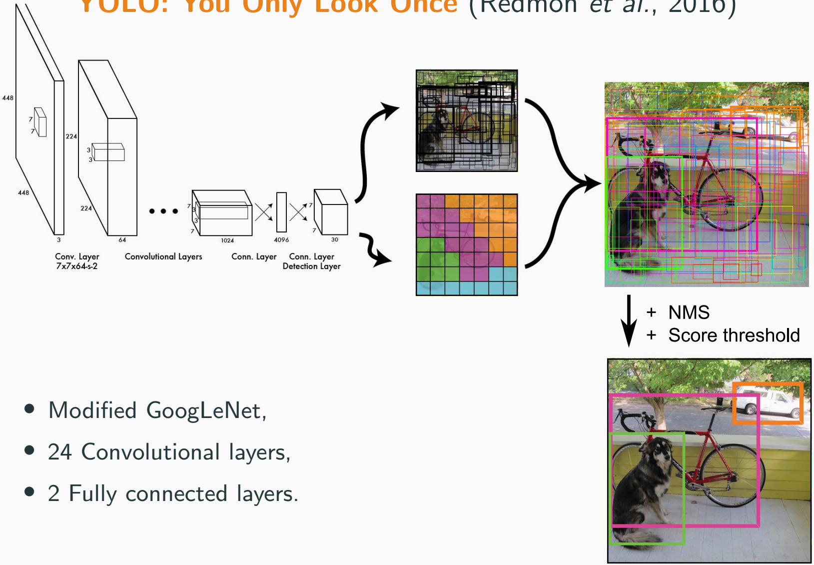
For Pascal VOC:

- 7x7 grid
- 2 bounding boxes / cell
- 20 classes



7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = **1470 outputs**

YOLO: You Only Look Once (Redmon *et al.*, 2016)



- Modified GoogLeNet,
- 24 Convolutional layers,
- 2 Fully connected layers.

(Multi-)Object detection

YOLO – Loss

$$\left.\begin{aligned} E(\boldsymbol{W}) = \lambda_{\text{coord}} & \sum_{i=1}^{7\times 7}\sum_{j=1}^{2} \mathbf{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_{ij})^2 + (y_i - \hat{y}_{ij})^2\right] \\ + \lambda_{\text{coord}} & \sum_{i=1}^{7\times 7}\sum_{j=1}^{2} \mathbf{1}_{ij}^{\text{obj}} \left[(\sqrt{w_i} - \sqrt{\hat{w}_{ij}})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_{ij}})^2\right] \\ + \quad & \sum_{i=1}^{7\times 7}\sum_{j=1}^{2} \mathbf{1}_{ij}^{\text{obj}} (c_i - \hat{c}_{ij})^2 + \lambda_{\text{noobj}} (1 - \mathbf{1}_{ij}^{\text{obj}}) \hat{c}_{ij}^2 \end{aligned}\right\} \begin{matrix} 7 \times 7 \times 2 \\ (x, y, w, h, c) \end{matrix}$$

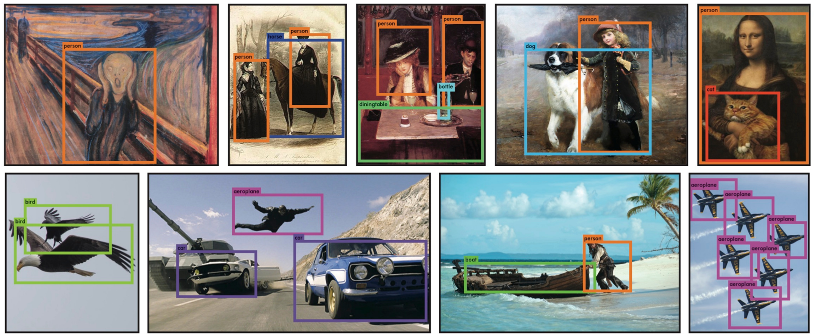
$$\left. + \quad \sum_{i=1}^{7\times 7} \mathbf{1}_{i}^{\text{obj}} \sum_{k=1}^{20} (p_{ik} - \hat{p}_{ik})^2 \right\} \begin{matrix} 7 \times 7 \times 20 \\ \hat{p} \end{matrix}$$

- $\mathbf{1}_{ij}^{\text{obj}}$ encodes if there is a j-th object in cell i,
- $\mathbf{1}_{i}^{\text{obj}}$ encodes if there is at least one object in cell i,
- λ_{coord}, λ_{noobj} controls the balance of the different terms.

Note: This is a reinterpretation of the loss written in the original paper.

YOLO – Results

- Slightly worse than Fast R-CNN on specific challenges,
- Better at generalizing on unseen datasets.



Qualitative Results. YOLO running on sample artwork and natural images from the internet. Made one mistake only, find it!

- Real-time: about 45 frames per second (Faster R-CNN is about 7fps),
- Demo: `http://pjreddie.com/yolo/`

YOLO v2 (Redmon & Farhadi, 2017)

- **dimension priors:** learn 5 anchors with k-means, instead of hand-picked ones,
- **location prediction:** parameterize the bbox s.t. its center is always in its cell,
- **passthrough:** add a shortcut connection (similar to SSD, ResNet, DenseNet),
- **multi-scale:** change input size during training (since fully convolutional).

	YOLO								YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓				
new network?					✓	✓	✓	✓	✓
dimension priors?						✓	✓	✓	✓
location prediction?						✓	✓	✓	✓
passthrough?							✓	✓	✓
multi-scale?								✓	✓
hi-res detector?									✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

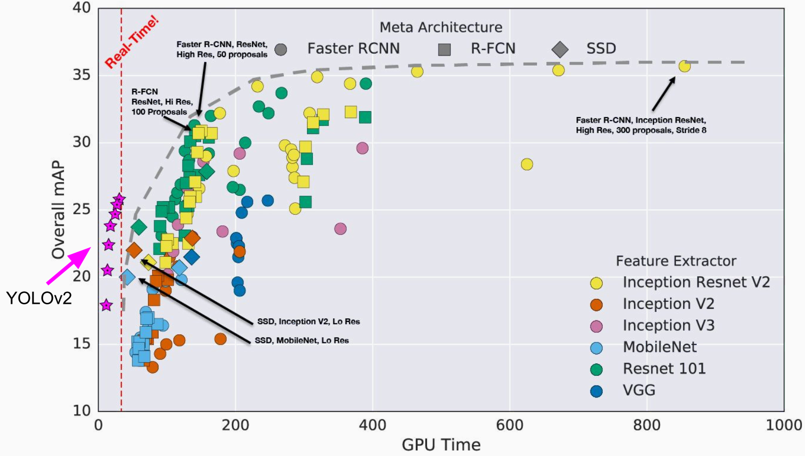
The path from YOLO to YOLOv2.

YOLO v2 – Comparisons

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

Detection frameworks on PASCAL VOC 2007.

YOLO v2 – Comparisons

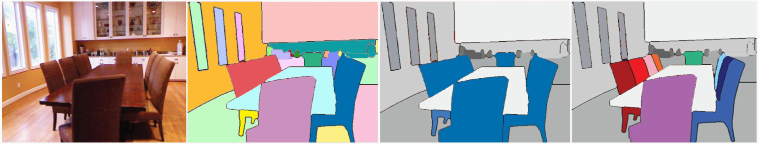


Huang, Jonathan, et al. "Speed/accuracy trade-offs for modern convolutional object detectors." *arXiv preprint arXiv:1611.10012* (2016).

CVPR'2017 Best Paper Honorable Mention Award

Segmentation

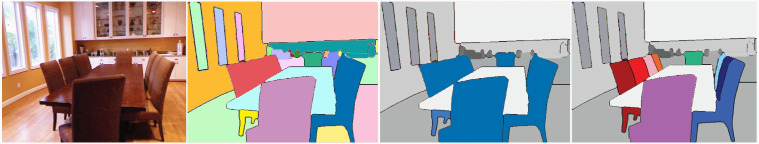
Segmentation – Terminology



Input Image | Segmentation | Semantic Segmentation | Semantic Instance Segmentation

- **Segmentation:**
  - Partition of an image into several "coherent" parts/segments,
  - Without any attempt at understanding what these parts represent,
  - Typically based on color, textures, smoothness of boundaries,
  - Also referred to as super-pixel segmentation.

Segmentation – Terminology



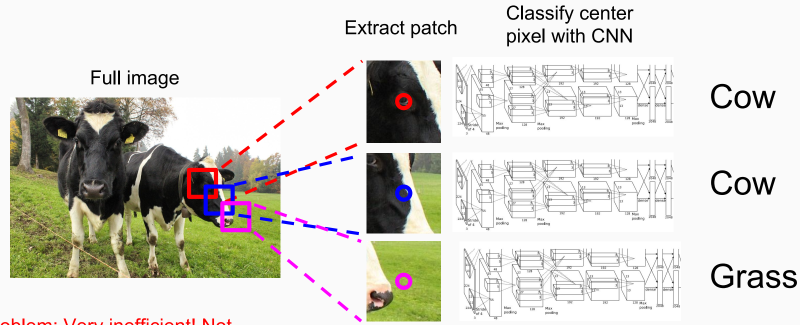
Input Image | Segmentation | Semantic Segmentation | Semantic Instance Segmentation

- **Semantic segmentation:**
    - Each segment corresponds to a class label (objects + background),
    - Also referred to as scene parsing or scene labeling.
- **Instance segmentation:**
    - Find object boundaries between objects, including delineations between instances of the same object.
- **Semantic instance segmentation:** find object boundaries + labels.

Semantic segmentation – Sliding window

(Farabet *et al.*, 2013, Pinheiro and Collobert, 2014)

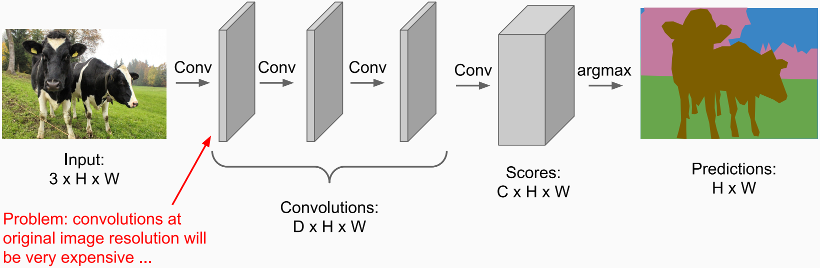
- Slide a window and predict the object class for each of them,
- Affect the class to the corresponding central pixel.



Problem: Very inefficient! Not reusing shared features between overlapping patches

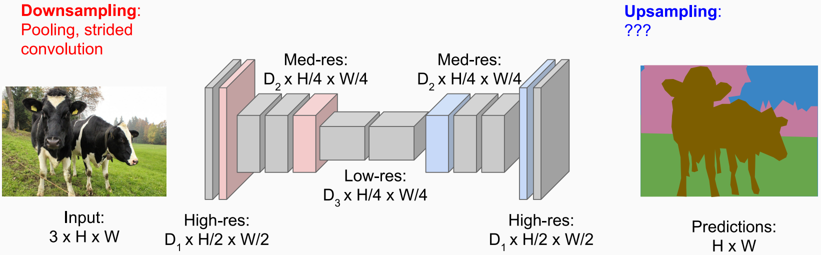
Semantic segmentation – Fully convolutional

- Design a network as a bunch of convolutional layers,
- Make predictions for all pixels all at once.



Semantic segmentation – Fully convolutional

- Design a network as a bunch of convolutional layers,
- Perform **downsampling** and **upsampling** inside the network.



Semantic segmentation – Unpooling

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

…

Rest of the network

Max Unpooling
Use positions from pooling layer

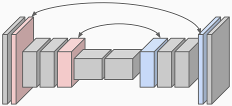
1	2
3	4

Input: 2 x 2

0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

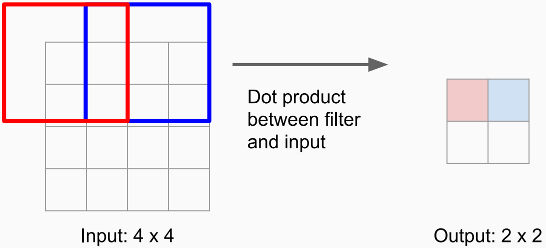
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Semantic segmentation – Transposed convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



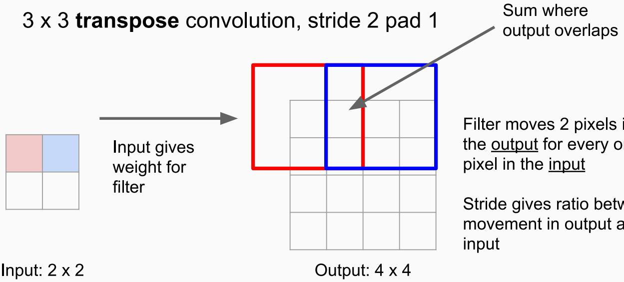
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Semantic segmentation – Transposed convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Filter moves 2 pixels in the <u>output</u> for every one pixel in the <u>input</u>

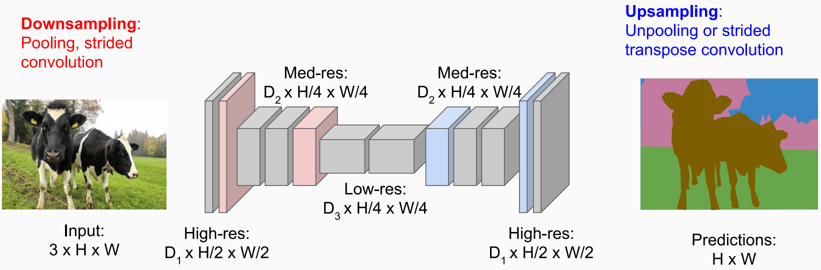
Stride gives ratio between movement in output and input

Also known as: deconvolutions (bad name) or fractionally strided convolutions.

Semantic segmentation – Overview

(Long *et al.*, 2015 & Noh *et al*, 2015)

- Design a network as a bunch of convolutional layers,
- Perform **downsampling** and **upsampling** inside the network.

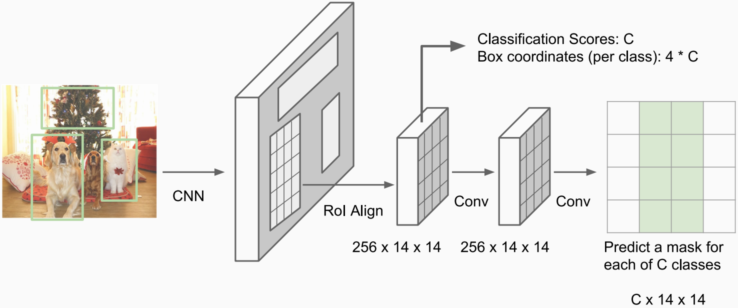


Problem: the two cows are merged together.

How to find boundaries between objects?

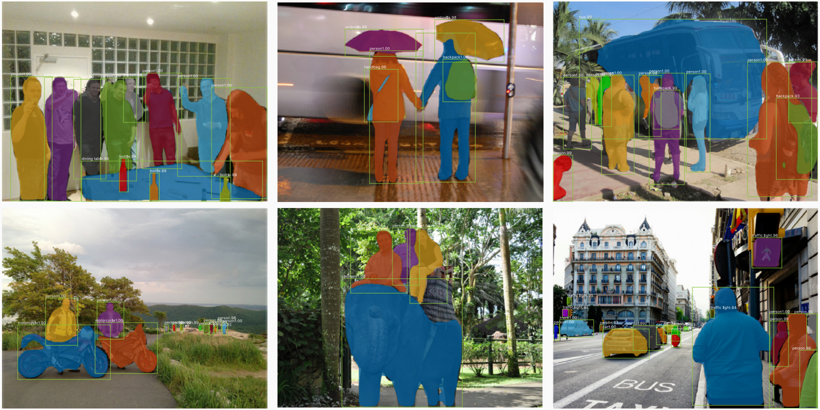
Instance segmentation – Mask R-CNN (He *et al.*, 2017)

- Perform instance segmentation and object detection jointly,
- Add a parallel branch to Faster R-CNN in order to predict an object mask,
- For each RoI, use one binary mask per class defined on a 14×14 grid,



- Each cell indicates if it is covered by the object of the given class,
- Learn the three tasks jointly: classification, bbox and mask prediction,
- At test time, combine results obtained at different scales.

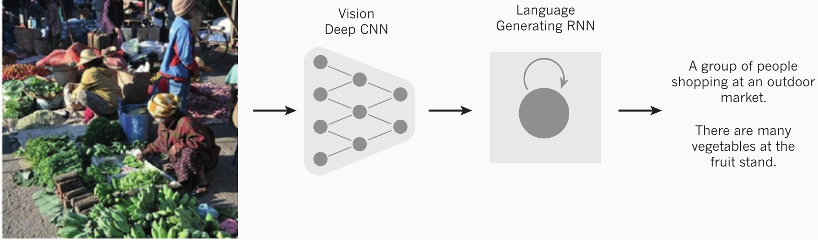
Instance segmentation – Mask R-CNN – Results



Provides really good results at about 5fps.

Image captioning

Image captioning



Goal: Generate fitting natural-language captions only based on the pixels.

How: Combine a vision deep CNN and a language-generating RNN.

What are Recurrent Neural Networks (RNNs)?

(Source: Lucas Masuch & Caner Hazırbaş)

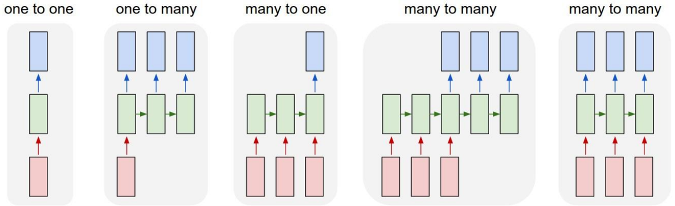
Recurrent Neural Networks (RNNs)

- Recurrent Neural Networks (RNNs) are Artificial Neural Networks that can deal with sequences of variable size.
- RNNs have a feedback loop where the net's output is fed back into the net along with the next input.
- RNNs receive an input and produce an output. Unlike other nets, the inputs and outputs can come in a sequence.
- Variant of RNN is Long Short Term Memory (LSTM).

State-of-the-art results in time series prediction: speech recognition, stock market prediction, language translation, language generation and other sequence learning problems. *Everything that can be processed sequentially.*

(Source: Caner Hazırbaş)

Recurrent Neural Networks (RNNs)



RNNs are general computers which can learn algorithms to map input sequences to output sequences (flexible-sized vectors). The output vector's contents are influenced by the entire history of inputs.

Examples:

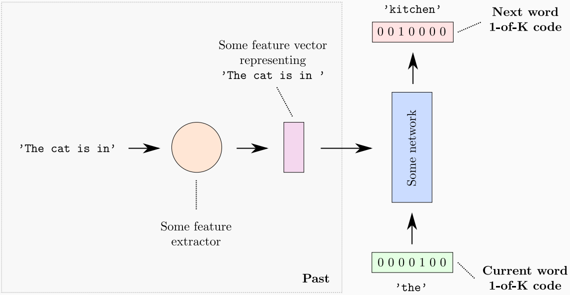
- one-to-one: image classification (traditional),
- one-to-many: image captioning,
- many-to-one: video classification,
- many-to-many: text translation, frame-by-frame classif.

(Source: Lucas Masuch)

Language generating RNNs – Training

How to learn `'The cat is in the kitchen drinking milk.'`?

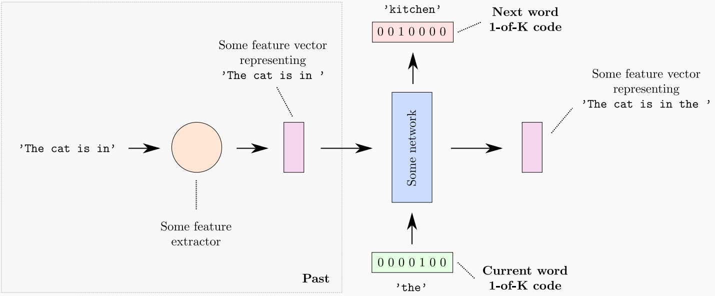
- Word: a 1-of-K code (large dictionary of K words),
- Learn: $\mathbb{P}$(next word | current word & past),
- Represent the past as a feature vector.



Language generating RNNs – Training

How to learn `'The cat is in the kitchen drinking milk.'`?

- Word: a 1-of-K code (large dictionary of K words),
- Learn: $\mathbb{P}$(next word | current word & past),
- Represent the past as a feature vector.

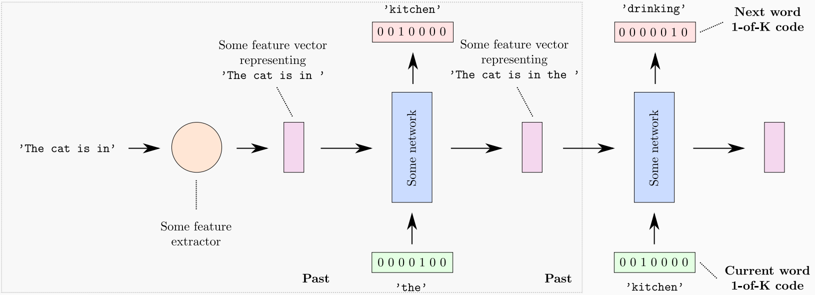


- Learn also how to represent the current sentence,

Language generating RNNs – Training

How to learn `'The cat is in the kitchen drinking milk.'`?

- Word: a 1-of-K code (large dictionary of K words),
- Learn: $\mathbb{P}$(next word | current word & past),
- Represent the past as a feature vector.

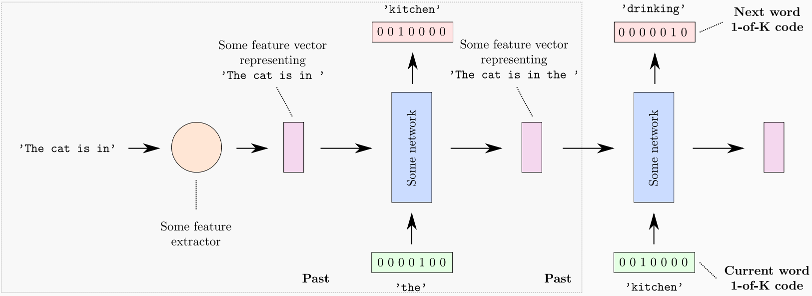


- Learn also how to represent the current sentence,
- Repeat for the next word,

Language generating RNNs – Training

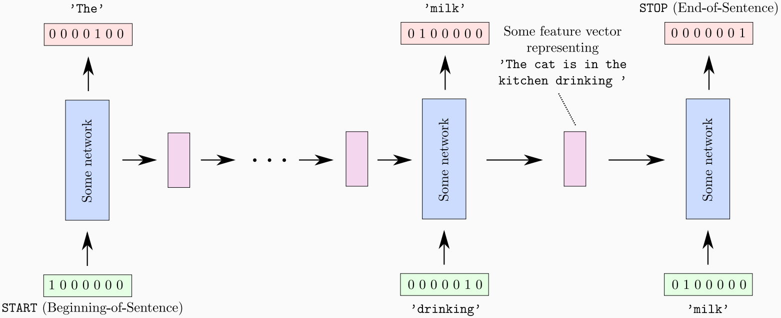
How to learn `'The cat is in the kitchen drinking milk.'`?

- Word: a 1-of-K code (large dictionary of K words),
- Learn: $\mathbb{P}$(next word | current word & past),
- Represent the past as a feature vector.



- Learn also how to represent the current sentence,
- Repeat for the next word, and the previous words.

Language generating RNNs – Training

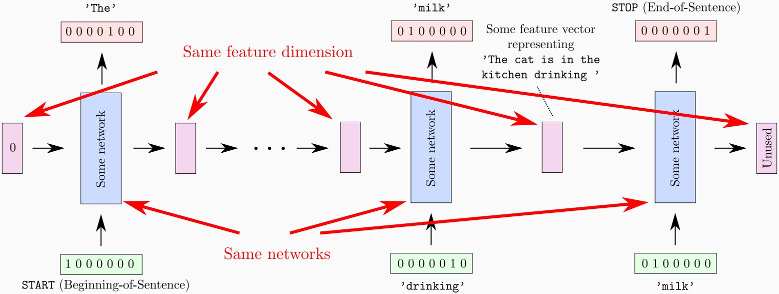


- Add two words: START and STOP to delimitate the sentence,
- Learn everything end-to-end on a large corpus of sentences,
- Minimize the sum of the cross-entropy of each word (maximum likelihood),
- Intermediate feature will learn how to memorize the past/context/state.

How should the network architecture and size of intermediate features evolve with the location in the sequence?

Language generating RNNs – Training

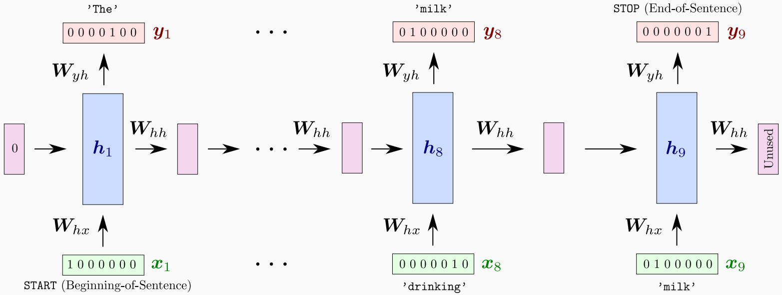
- Use the same networks and the same feature dimension,
- The past is always embedded in a fix-sized feature,
- Set the first feature as a zero tensor.



- Allows you to learn from arbitrarily long sequences,
- Sharing the architecture ⇒ less parameters ⇒ training requires less data and the final prediction can be expected to be more accurate.

Language generating RNNs – Training

Example of training a simple shallow RNN

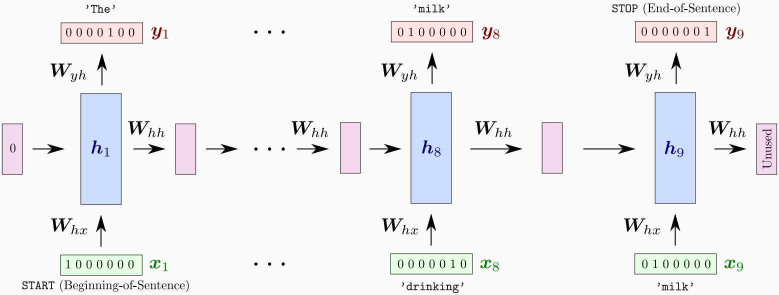


$$\boldsymbol{h}_t = g(\boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

Language generating RNNs – Training

Example of training a simple shallow RNN



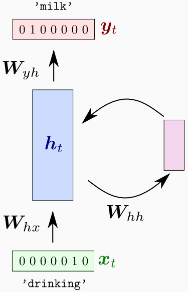
Unfolded representation of the RNN for a fixed-length sequence.

$$\boldsymbol{h}_t = g(\boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

Language generating RNNs – Training

Example of training a simple shallow RNN



Unfolded representation of the RNN for a fixed-length sequence.

Folded representation: A RNN is nothing else than an ANN with loops.

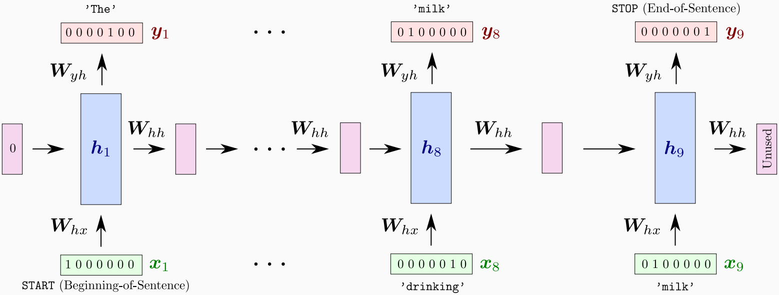
$$\boldsymbol{h}_t = g(\boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

Image captioning – Recurrent Neural Networks (RNNs)

Language generating RNNs – Training

Example of training a simple shallow RNN



Unfolded representation of the RNN for a fixed-length sequence.

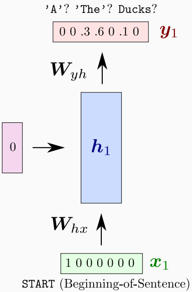
Folded representation: A RNN is nothing else than an ANN with loops.

$$\boldsymbol{h}_t = g(\boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \text{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

Language generating RNNs – Testing

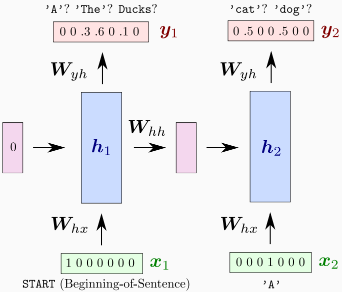
Example of generating sentences from a simple shallow RNN



- Provide START, get all the probabilities $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{START})$,

Language generating RNNs – Testing

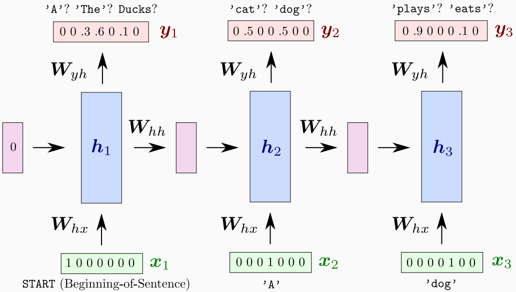
Example of generating sentences from a simple shallow RNN



- Provide `START`, get all the probabilities $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{START})$,
- Select one of these words according to their probabilities, let say `'A'`,
- Provide `'A'` and the past, and get $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{'A'} \;\&\; \text{past})$,

Language generating RNNs – Testing

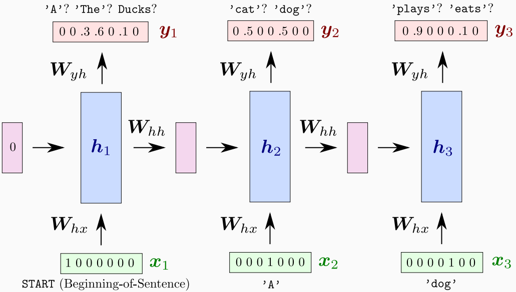
Example of generating sentences from a simple shallow RNN



- Provide `START`, get all the probabilities $\mathbb{P}$(next word | current word = `START`),
- Select one of these words according to their probabilities, let say `'A'`,
- Provide `'A'` and the past, and get $\mathbb{P}$(next word | current word = `'A'` & past),
- Repeat while generating the sentence `'A dog plays          '`

Language generating RNNs – Testing

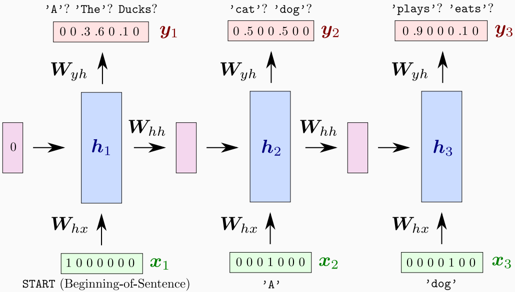
Example of generating sentences from a simple shallow RNN



- Provide START, get all the probabilities $\mathbb{P}$(next word | current word = START),
- Select one of these words according to their probabilities, let say 'A',
- Provide 'A' and the past, and get $\mathbb{P}$(next word | current word = 'A' & past),
- Repeat while generating the sentence 'A dog plays with   '

Language generating RNNs – Testing

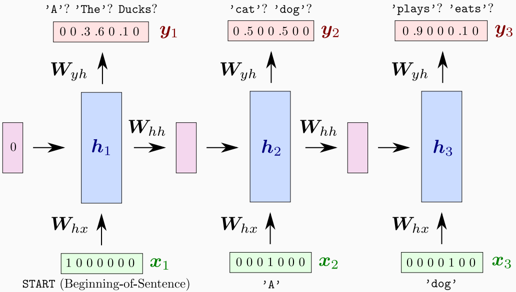
Example of generating sentences from a simple shallow RNN



- Provide `START`, get all the probabilities $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{START})$,
- Select one of these words according to their probabilities, let say `'A'`,
- Provide `'A'` and the past, and get $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{'A'} \;\&\; \text{past})$,
- Repeat while generating the sentence `'A dog plays with a      '`

Language generating RNNs – Testing

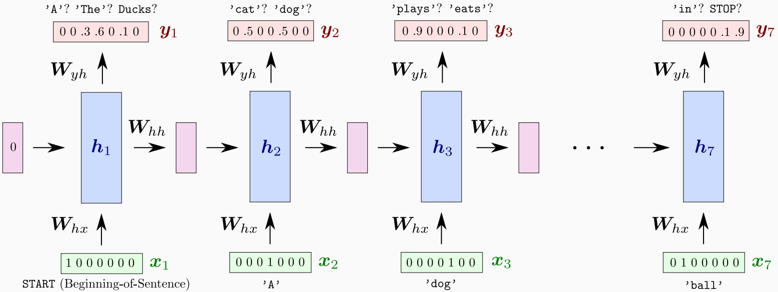
Example of generating sentences from a simple shallow RNN



- Provide `START`, get all the probabilities $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{START})$,
- Select one of these words according to their probabilities, let say `'A'`,
- Provide `'A'` and the past, and get $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{'A'}\ \&\ \text{past})$,
- Repeat while generating the sentence `'A dog plays with a ball'`

Language generating RNNs – Testing

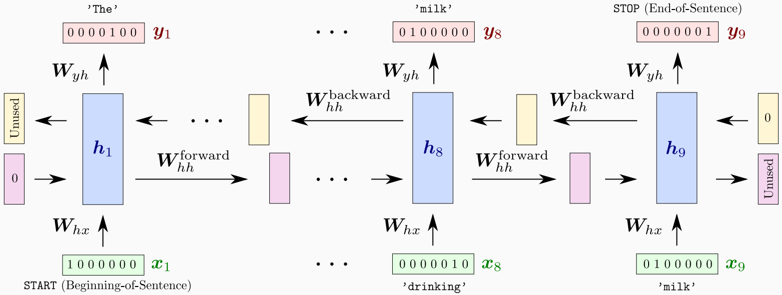
Example of generating sentences from a simple shallow RNN



- Provide `START`, get all the probabilities $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{START})$,
- Select one of these words according to their probabilities, let say `'A'`,
- Provide `'A'` and the past, and get $\mathbb{P}(\text{next word} \mid \text{current word} = \texttt{'A'} \;\&\; \text{past})$,
- Repeat while generating the sentence `'A dog plays with a ball'`
- Stop as soon as you have picked `STOP`.

Language generating RNNs – Other architectures

Example of a bidirectional RNN



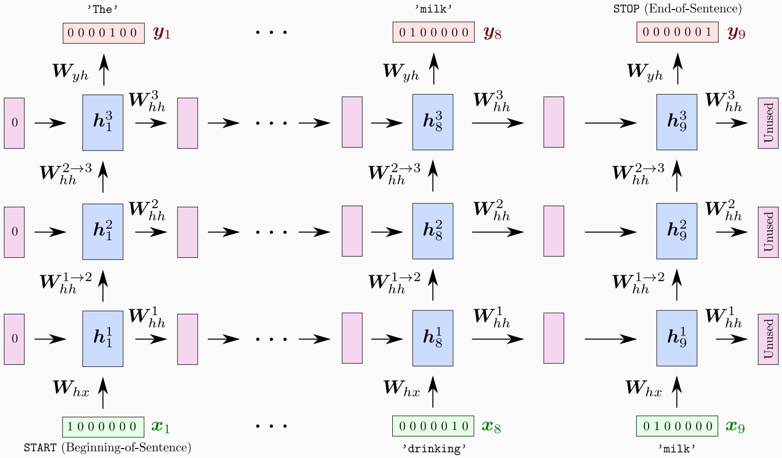
Output at time t may not only depend on the previous elements, but also on **future elements**.

$$\boldsymbol{h}_t = g(\boldsymbol{W}_{hx}\boldsymbol{x}_t + \boldsymbol{W}_{hh}^{\text{forward}}\boldsymbol{h}_{t-1} + \boldsymbol{W}_{hh}^{\text{backward}}\boldsymbol{h}_{t+1} + \boldsymbol{b}_h)$$

$$\boldsymbol{y}_t = \text{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

Language generating RNNs – Other architectures

Example of a deep RNN with 3 hidden layers

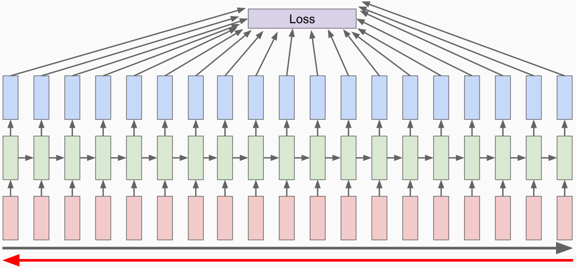


We now have multiple layers per time step (a feature hierarchy).
Higher learning capacity but requires a lot more training data.

Language generating RNNs – Learning algorithm

Backpropagation through time (BPTT)

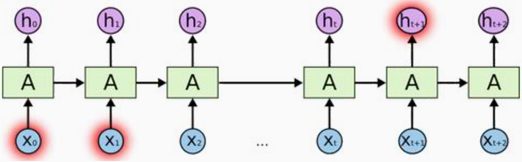
- Similar to standard backprop for training a traditional Neural Network,
- During training, unfold the network to the size of each training sequence,
- Take into account that parameters are shared by all steps in the network.



Forward through the entire sequence to compute the loss,
then backward through entire sequence to compute gradients.

Language generating RNNs – Limitations

- Vanilla RNNs have difficulties learning long-term dependencies,



'I grew up in France... I speak fluent ???'

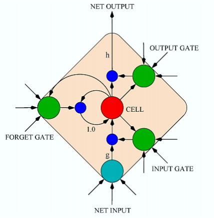
→ We need the context of France from further back.

- One reason is again the vanishing/exploding gradient problems,
- Certain types of RNNs are specifically designed to get around them.

→ **Long-Short-Term Memory (LSTM)**

Long Short-Term Memory RNN (LSTM)

(Hochreiter & Schmidhuber, 1997)



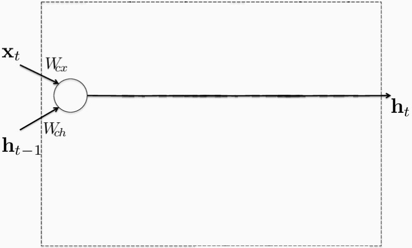
A Long Short-Term Memory (LSTM) network is a particular type of recurrent network that works slightly better in practice, owing to its more powerful update equation and some appealing back propagation dynamics.

- The **LSTM units** give the network memory cells with read, write and reset operations. During training, the network can learn when it should remember data and when it should throw it away.
- Well-suited to learn from experience to classify, process and predict time series when there are very long time lags of unknown size between important events.

Long Short-Term Memory RNN (LSTM)

$$\boldsymbol{h}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \qquad \leftarrow \text{memory}$$

$$\boldsymbol{y}_t = \text{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y) \qquad \leftarrow \text{used as feature for prediction}$$



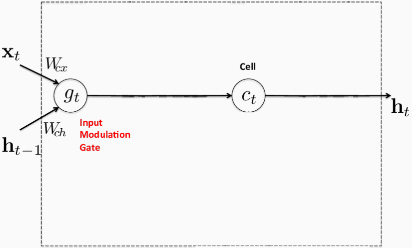
Long Short-Term Memory RNN (LSTM)

$$\boldsymbol{g}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \qquad \leftarrow \text{input modulation gate}$$

$$\boldsymbol{c}_t = \boldsymbol{g}_t \qquad \leftarrow \text{place memory in a cell unit } \boldsymbol{c}$$

$$\boldsymbol{h}_t = \boldsymbol{c}_t$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y) \qquad \leftarrow \text{but use } \boldsymbol{h}_t \text{ to make prediction}$$



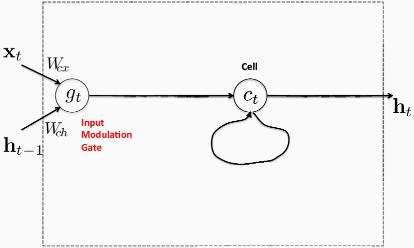
Long Short-Term Memory RNN (LSTM)

$$\boldsymbol{g}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \quad \leftarrow \text{input modulation gate}$$

$$\boldsymbol{c}_t = \boldsymbol{c}_{t-1} + \boldsymbol{g}_t \quad \leftarrow \text{the cell keeps track of long term}$$

$$\boldsymbol{h}_t = \boldsymbol{c}_t$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$



Long Short-Term Memory RNN (LSTM)

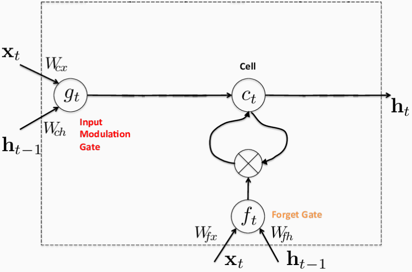
$$\boldsymbol{f}_t = \mathrm{sigm}(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \qquad \leftarrow \text{forget gate}$$

$$\boldsymbol{g}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \qquad \leftarrow \text{input modulation gate}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} + \boldsymbol{g}_t \qquad \leftarrow \text{but can forget some of its memories}$$

$$\boldsymbol{h}_t = \boldsymbol{c}_t \qquad (\otimes = \text{element wise product})$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$



Long Short-Term Memory RNN (LSTM)

$$\boldsymbol{i}_t = \mathrm{sigm}(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \qquad \leftarrow \text{input gate}$$

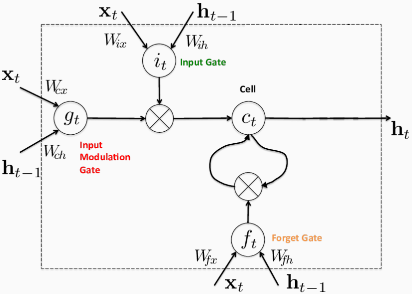
$$\boldsymbol{f}_t = \mathrm{sigm}(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \qquad \leftarrow \text{forget gate}$$

$$\boldsymbol{g}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \qquad \leftarrow \text{input modulation gate}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \otimes \boldsymbol{g}_t \qquad \leftarrow \text{and ignore some of the update}$$

$$\boldsymbol{h}_t = \boldsymbol{c}_t$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$



Long Short-Term Memory RNN (LSTM)

$$\boldsymbol{o}_t = \mathrm{sigm}(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o)$$ ← output gate

$$\boldsymbol{i}_t = \mathrm{sigm}(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i)$$ ← input gate

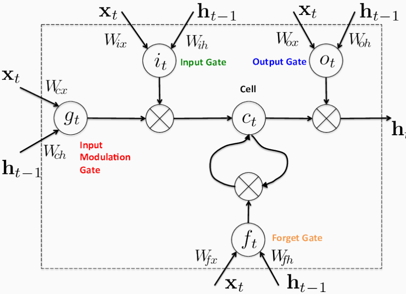
$$\boldsymbol{f}_t = \mathrm{sigm}(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f)$$ ← forget gate

$$\boldsymbol{g}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c)$$ ← input modulation gate

$$\boldsymbol{c}_t = \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \otimes \boldsymbol{g}_t$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \otimes \boldsymbol{c}_t$$ ← weight memory for generating feature

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$



Long Short-Term Memory RNN (LSTM)

$$\boldsymbol{o}_t = \mathrm{sigm}(\boldsymbol{W}_{ox}\boldsymbol{x}_t + \boldsymbol{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \qquad \leftarrow \text{output gate}$$

$$\boldsymbol{i}_t = \mathrm{sigm}(\boldsymbol{W}_{ix}\boldsymbol{x}_t + \boldsymbol{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \qquad \leftarrow \text{input gate}$$

$$\boldsymbol{f}_t = \mathrm{sigm}(\boldsymbol{W}_{fx}\boldsymbol{x}_t + \boldsymbol{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \qquad \leftarrow \text{forget gate}$$

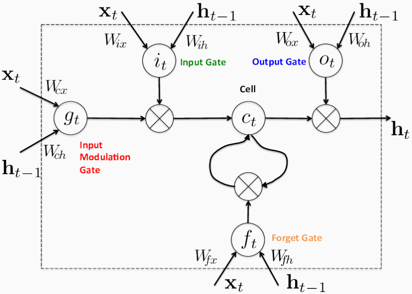
$$\boldsymbol{g}_t = g(\boldsymbol{W}_{cx}\boldsymbol{x}_t + \boldsymbol{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \qquad \leftarrow \text{input modulation gate}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \otimes \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \otimes \boldsymbol{g}_t$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \otimes \boldsymbol{c}_t \qquad \leftarrow \text{weight memory for generating feature}$$

$$\boldsymbol{y}_t = \mathrm{softmax}(\boldsymbol{W}_{yh}\boldsymbol{h}_t + \boldsymbol{b}_y)$$

There are many variants, but this is the general idea.



LSTM – Example of generated text

at first:

```
tyntd-iafhatawiaoihrdemot  lytdws  e ,tfti, astai f ogoh eoase rrranbyne 'nhthnee e
plia tklrgd t o idoe ns,smtt   h ne etie h,hregtrs nigtike,aoaenns lng
```

train more

```
"Tmont thithey" fomesscerliund
Keushey. Thom here
sheulke, anmerenith ol sivh I lalterthend Bleipile shuwy fil on aseterlome
coaniogennc Phe lism thond hon at. MeiDimorotion in ther thize."
```

train more

```
Aftair fall unsuch that the hall for Prince Velzonski's that me of
her hearly, and behs to so arwage fiving were to it beloge, pavu say falling misfort
how, and Gogition is so overelical and ofter.
```

train more

```
"Why do what that day," replied Natasha, and wishing to himself the fact the
princess, Princess Mary was easier, fed in had oftened him.
Pierre aking his soul came to the packs and drove up his father-in-law women.
```

Multi-layer LSTM trained on character sequences from texts by W. Shakespeare.

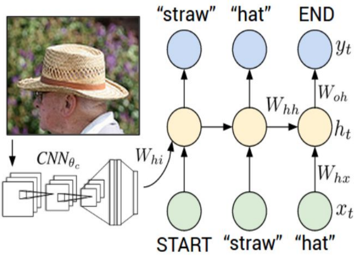
Further reading: `http://karpathy.github.io/2015/05/21/rnn-effectiveness/`

Image Captioning – Combining CNN and RNN

DeepImageSent (Karpathy *et al.*, CVPR 2015)

only takes into account image features
in the first hidden state

$$\begin{aligned} b_v &= W_{hi}[CNN_{\theta_c}(I)] \\ h_t &= f(W_{hx}x_t + W_{hh}h_{t-1} + b_h + \boxed{\mathbb{1}(t=1) \odot b_v}) \\ y_t &= softmax(W_{oh}h_t + b_o). \end{aligned}$$

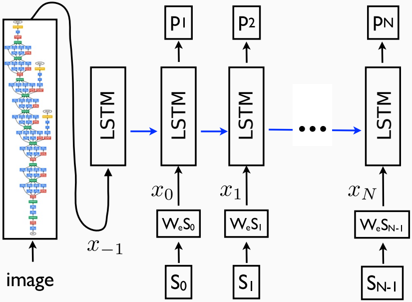


Multimodal Recurrent Neural Network

- Plug a standard CNN (its last feature layer) to a vanilla RNN,
- The CNN features are embedded to serve as initial memory state at $t = 1$,
- Perform end-to-end learning on a large corpus of captioned images,
- Words and images are automatically embedded in a common feature space.

Image Captioning – Combining CNN and RNN

Show and Tell (Vinyals *et al.*, CVPR 2015)



- Similar to DeepImageSent, but use LSTM instead of a vanilla RNN,
- Learn to embed words to the feature space of the CNN (role of W_e),
- The CNN features are used as input at $t = -1$.

Image Captioning – Combining CNN and RNN

Successful results

Captions generated using neuraltalk2
All images are CC0 Public domain: cat suitcase, cat tree, dog, bear, surfers, tennis, giraffe, motorcycle

A cat sitting on a suitcase on the floor

A cat is sitting on a tree branch

A dog is running in the grass with a frisbee

A white teddy bear sitting in the grass

Two people walking on the beach with surfboards

A tennis player in action on the court

Two giraffes standing in a grassy field

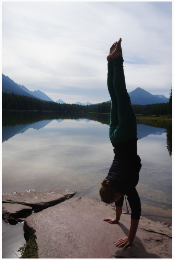
A man riding a dirt bike on a dirt track

Image Captioning – Combining CNN and RNN

Failure results

A woman is holding a cat in her hand

A person holding a computer mouse on a desk



A woman standing on a beach holding a surfboard

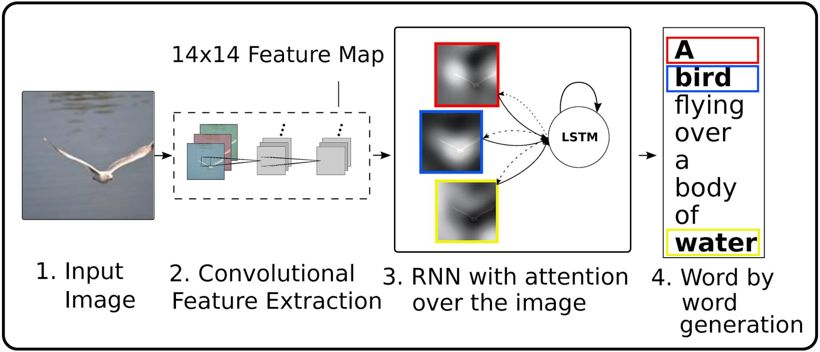
A bird is perched on a tree branch

A man in a baseball uniform throwing a ball

Captions generated using neuraltalk2

Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)



Force the RNN to focus its attention at a different spatial location when generating each word.

Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)

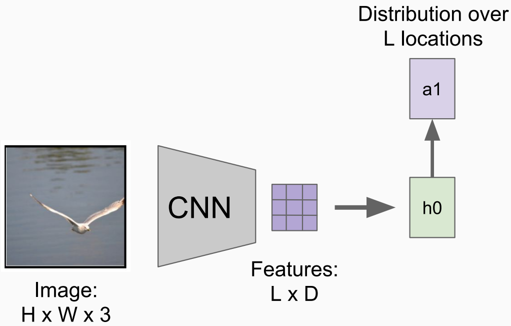


Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)

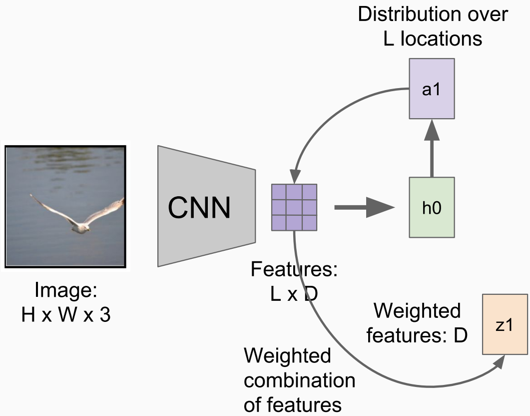


Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)

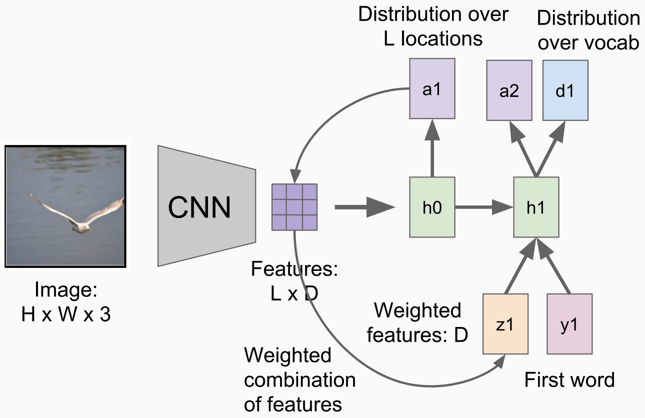


Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)

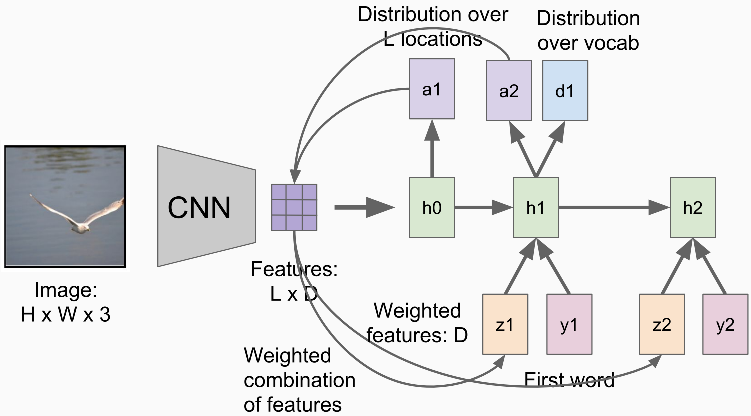


Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)

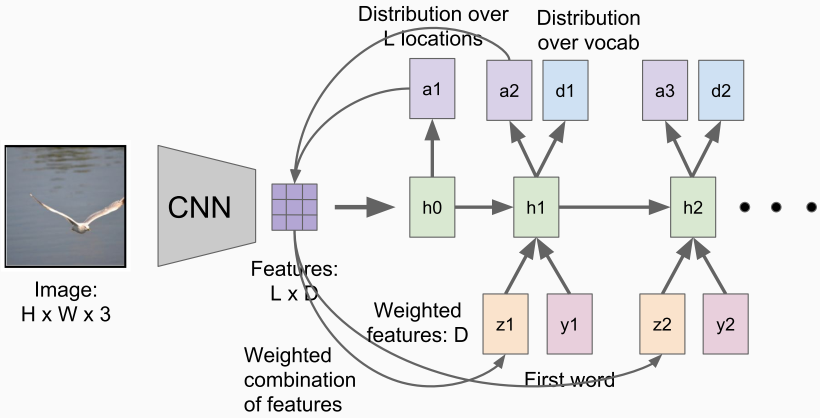


Image captioning – Combining CNN and RNN

Image Captioning – Combining CNN and RNN

Show, Attend and Tell (Xu *et al.*, 2015)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

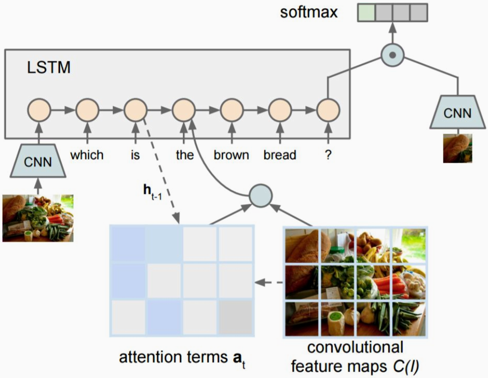
A little girl sitting on a bed with a teddy bear.

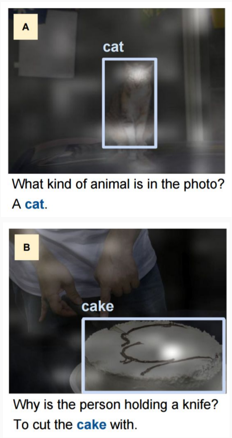
A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

Visual Question Answering: RNNs with Attention

(Zhu et al, 2016)





Visual Question Answering: RNNs with Attention

(Zhu et al, 2016)

Questions?

Next class: Generation, super-resolution and style transfer

Sources, images courtesy and acknowledgment

L. Araujo dos Santos
M. Bolaños
G. Chen
K. He
A. Horodniceanu
J. Johnson
A. Karpathy
F.-F. Li
R. Poisson
A. Salvador
S. Yeung