DEEP VISION IN ACTION

From classification and detection to segmentation

Duy V. Huynh, (NAIR) NewAI-VN Research

May 15, 2020

Outline

1. Recap: CNN Backbone Architecture

2. Segmentation and Detection


dont worry about it if you don't
understand

Recap: CNN Backbone Architecture

LeNet[1,2]

- Convolutional: locally connected, weight-sharing
- Subsampling
- Fully-connected layer
- Train by back-propagation

$\Longrightarrow$ Basic components of modern ConvNets!

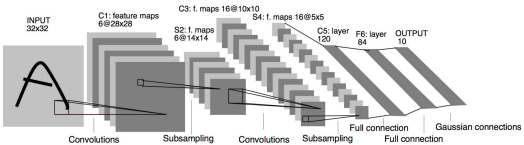


Figure 1: LeNet Architecture

[1] Yann Lecun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE*. 1998, pp. 2278–2324.

[2] Le Cun et al. "Handwritten Digit Recognition with a Back-Propagation Network". In: *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1990, pp. 396–404.

AlexNet[3]

LeNet-style backbone, and:

- ReLU[Nair & Hinton 2010]
  - Revolution of deep learning.
  - Accelerate trainning, better than backprop + tanh.
- Dropout[Hinton et al 2012]
  - In-network ensembling.
  - Reduce overfitting

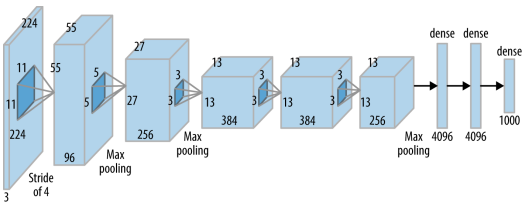


Figure 2: AlexNet Architecture

[3]Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. P. 2012.

VGG Net[4]

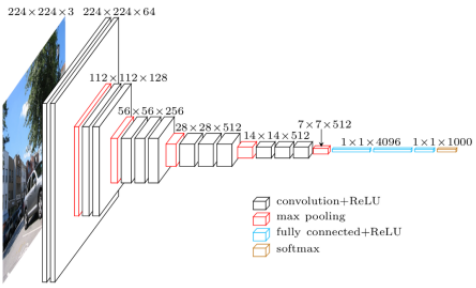


Figure 3: VGG-16 architecture

Modularized design:

- 3x3 Conv = 1 module
- Stack the same module
- Same computation for each module (1/2 spatial size $\Longrightarrow$ 2x filters)

Stage-wise trainning

- VGG-11 $\Longrightarrow$ VGG-13 $\Longrightarrow$ VGG-16

[4]Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. 2014.

GoogleNet/Inception[5]

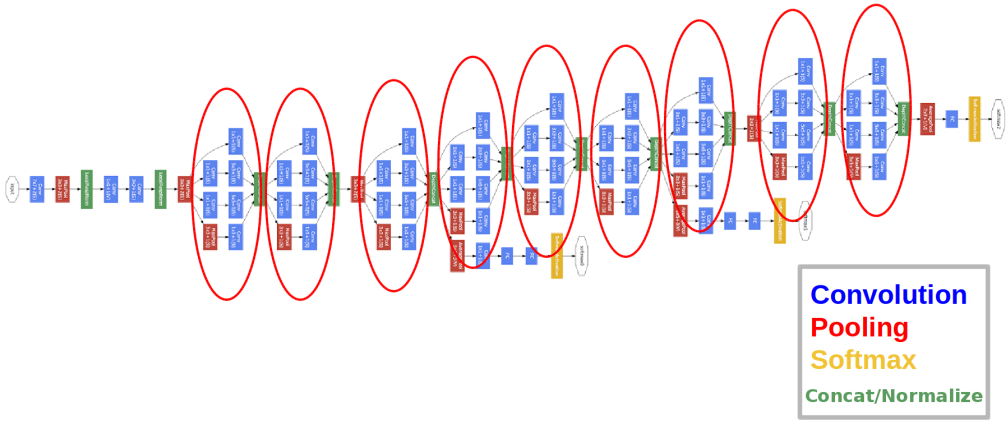


Figure 4: GoogleNet/Inception architecture

[5]Christian Szegedy et al. *Going deeper with convolutions*. 2014.

1×1 convolution[6]

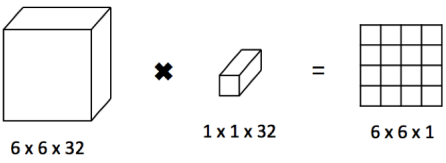


Figure 5: 1x1 conv with 1 filter

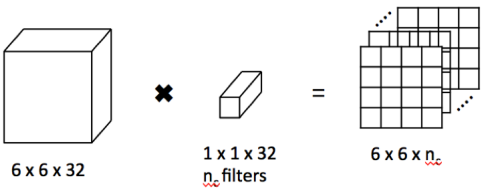


Figure 6: 1x1 conv with n filter

[6] Min Lin, Qiang Chen, and Shuicheng Yan. *Network in Network*. 2013.

Why 1×1 convolution?

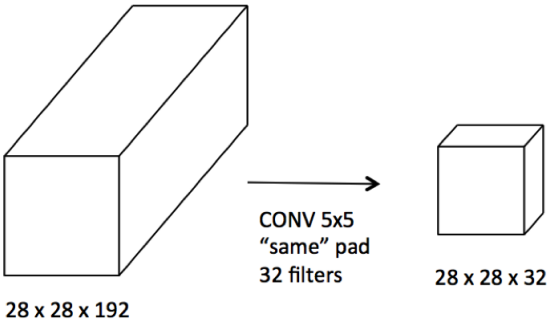


Figure 7: 5×5 filter ×32

$\Longrightarrow$ Computational cost: $(28 \times 28 \times 32) \times (5 \times 5 \times 192) = 120.4M$

Why 1×1 convolution?

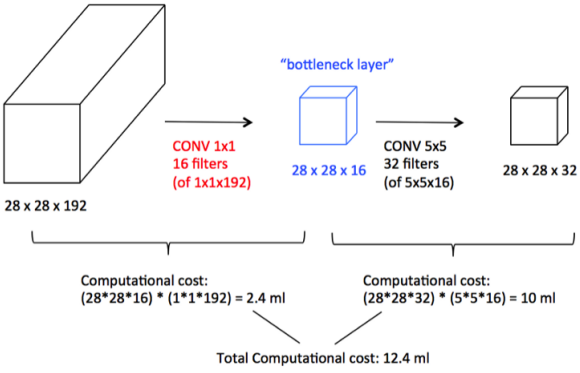


Figure 8: "Bottleneck" transfer

$\Longrightarrow$ 1×1 reduces by nearly 10 times the total computational cost.

Inception Module

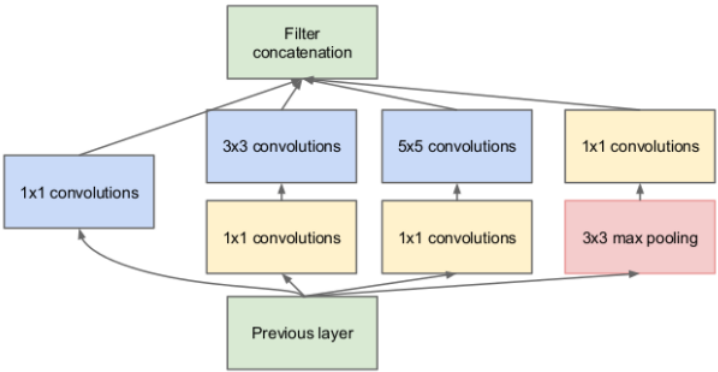


Figure 9: Inception module

GoogleNet/Inception[7]

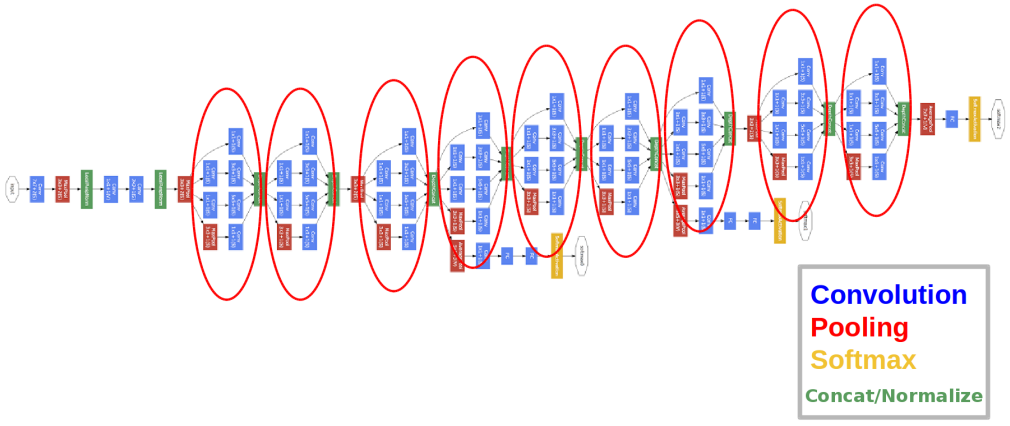


Figure 10: GoogleNet/Inception architecture

[7] Christian Szegedy et al. *Going deeper with convolutions*. 2014.

GooleNet/Inception[8]

Conclusions

- Multiple "branches"
  - e.g., 1x1 conv, 3x3 conv, 5x5 conv, pool, ...
- Shorcuts
  - stand-alone 1x1, merged by concatenation method.
- Bottlenek
  - Reduce dimensionality by 1x1 conv.

[8] Christian Szegedy et al. *Going deeper with convolutions*. 2014.


I WAS WINNING
IMAGENET
UNTIL A
DEEPER MODEL
CAME ALONG

Stacking more and more layers?[9]

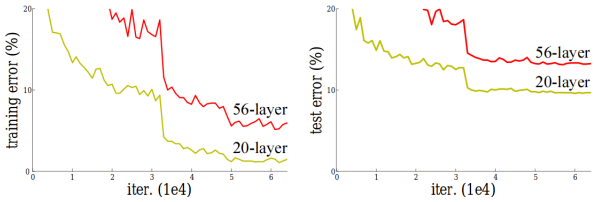
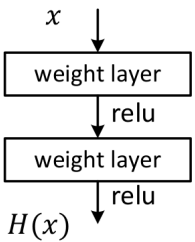


Figure 11: Training error (left) and test error (right) on CIFAR-10 with 20-layer and 56-layer plain networks

[9] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

Deep Residual Learning[10]



$H(x)$ is any desired mapping, hope the small subnet fit $H(x)$.

Figure 12: "Plain" connection in "plain" net

[10]Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

Deep Residual Learning[11]

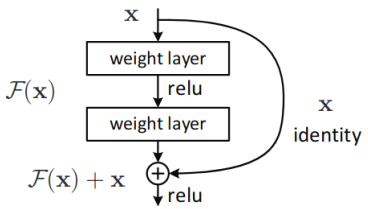


Figure 13: Residual connection in residual net

$H(x)$ is any desired mapping, ~~hope the small subnet fit $H(x)$~~ hope the small subnet fit $F(x)$ let $H(x) = F(x) + x$.

- If identity were optimal, easy to set weights as 0.
- If optimal mapping is closer to identity, easier to find small fluctuations.

[11] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

ImageNet Experiments

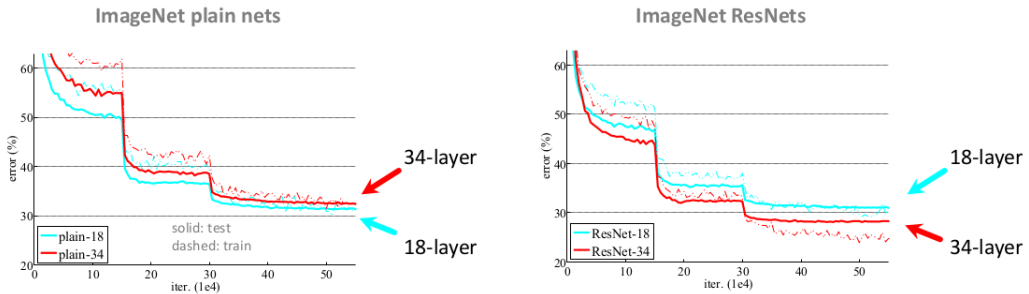


Figure 14: Training on ImageNet

- Deep ResNets can be trained without any difficulties.
- Deeper ResNets have lower trainning error, and also lower test error.

ImageNet Experiments

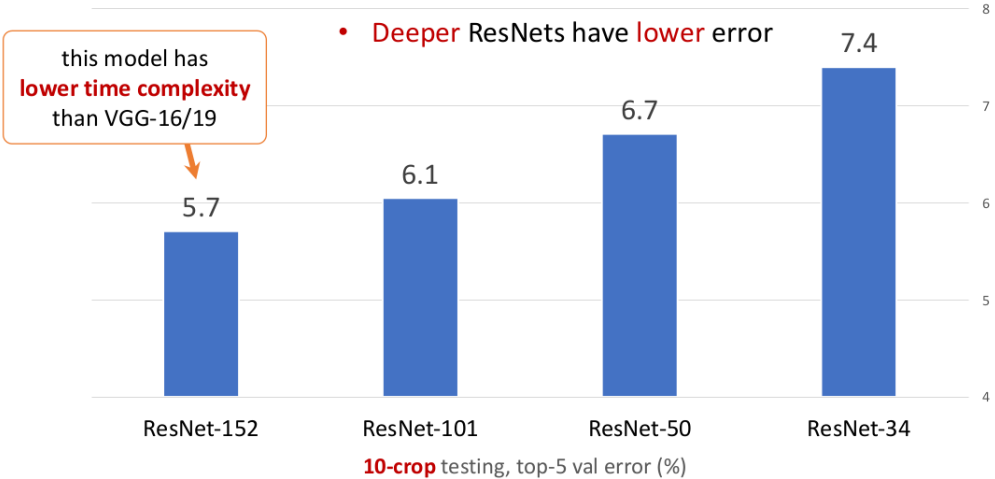


Figure 15: Training more and more deeper model on ImageNet

Residual connection beyond Computer Vision

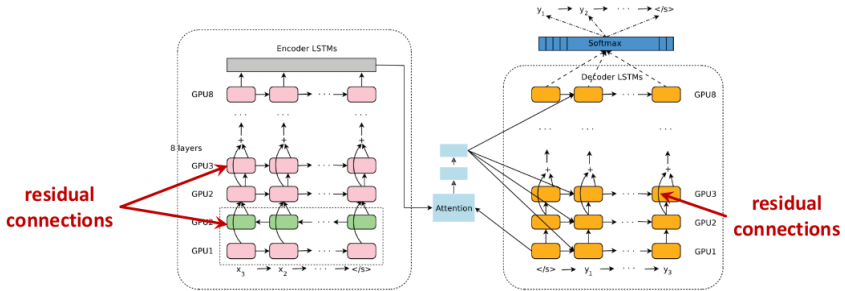


Figure 16: Neural Machine Translation (NMT)[12]: 8-layer LSTM

[12]Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016).

Residual connection beyond Computer Vision

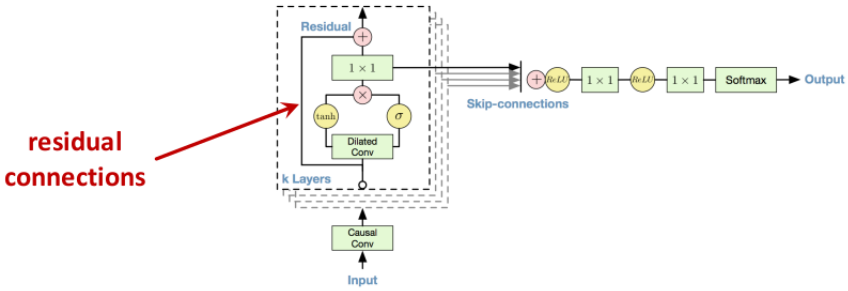


Figure 17: Speech synthesis (WaveNet)[13]: Residual CNNs on 1-d sequence

[13] Aäron van den Oord et al. "WaveNet: A Generative Model for Raw Audio". In: *SSW*. 2016.

Residual connection beyond Computer Vision

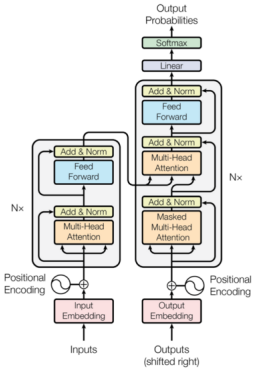


Figure 18: Transformer[14]:

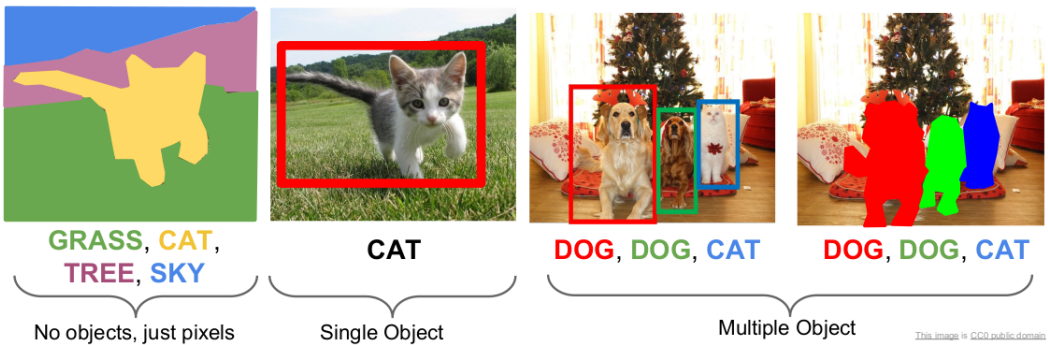
[14] Ashish Vaswani et al. "Attention Is All You Need". In: *NIPS*. 2017.

More architectures

- **Inception-ResNet**[Szegedy et al 2017]: Inception as transformation + residual connection.
- **DenseNet**[Huang et al CVPR 2017]: Densely connected shortcuts w/ concat.
- **Xception**[Chollet CVPR 2017], **MobileNets**[Howard et al 2017]: DepthwiseConv
- **ShuffleNet**[Zhang et al 2017]: More group/DepthwiseConv + shuffle
- **ResNeXt**[He et al CVPR 2017]: Residual connection, multi branch, group-conv
- ...

Segmentation and Detection

Computer Vision is not only Classification



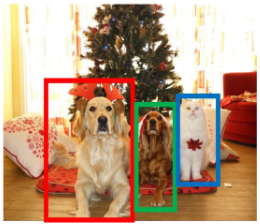
Computer Vision is not only Classification

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

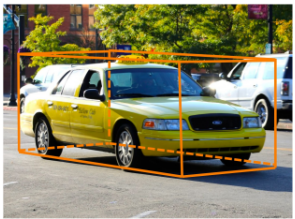
2D Object Detection



DOG, DOG, CAT

Object categories + 2D bounding boxes

3D Object Detection

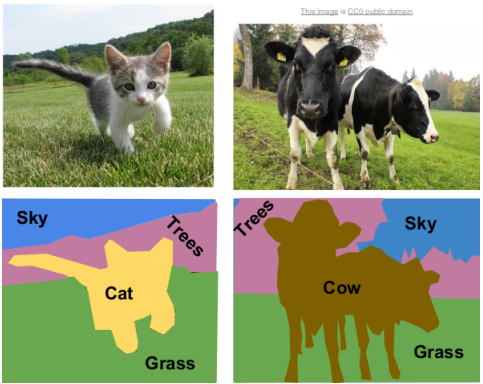


Car

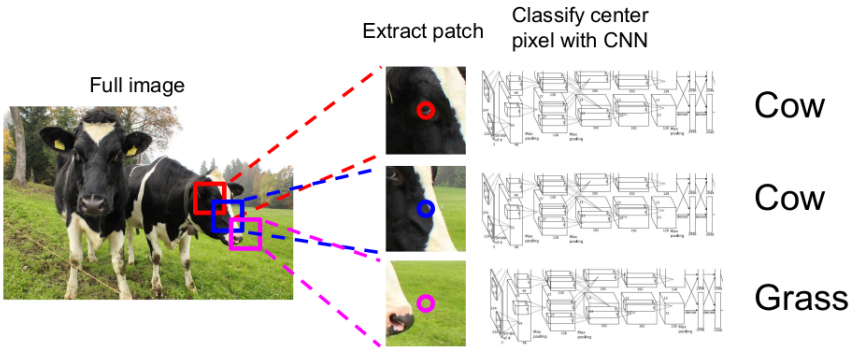
Object categories + 3D bounding boxes

Semantic Segmentation

- Label each pixel in the image with a category label.
- Don't differentiate instances, only care about pixels.

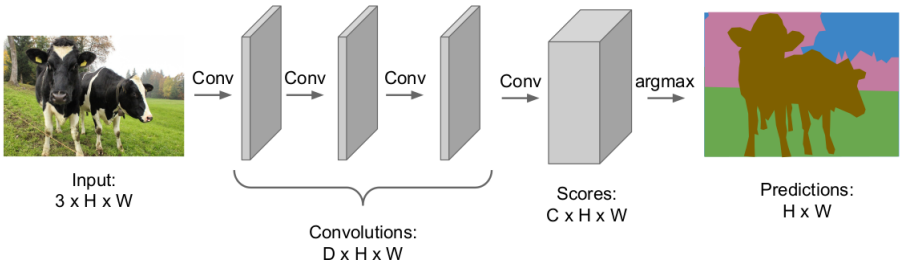


Semantic Segmentation Idea: Sliding Window



$\Longrightarrow$ **Problem**: Computational cost is very expensive! Not reusing shared features between overlapping patches.

Semantic Segmentation Idea: Fully Convolutional



- Make predictions for all pixels at once

$\Longrightarrow$ **Problem** again: Convolutions at original image resolution is still very expensive.

Semantic Segmentation Idea: Fully Convolutional

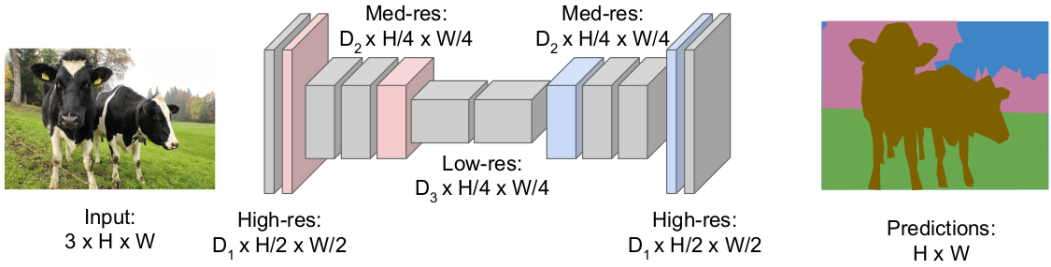
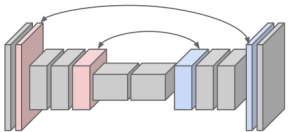


Figure 19: Fully Convolutional Networks for Semantic Segmentation

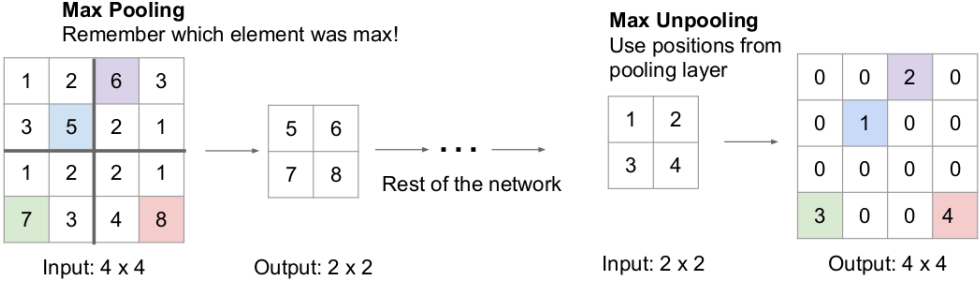
Semantic Segmentation Idea: Fully Convolutional



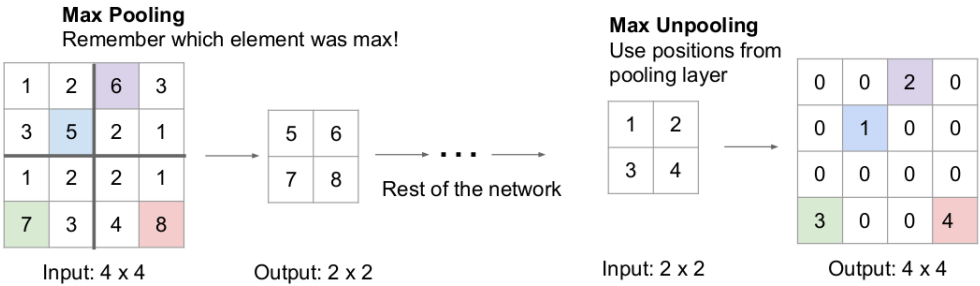
- Design network as a bunch of convolutional layers, with **down-sampling** and **up-sampling** inside[15].
- Down-sampling: Pooling, strided convolution $\Longrightarrow$ Up-sampling, Un-pooling ???

[15] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3431–3440.

Naive Idea for Up-sampling



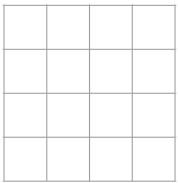
Naive Idea for Up-sampling



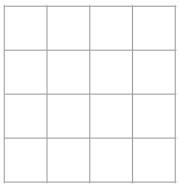
- Nonlearnable $\Longrightarrow$ Not a very good idea.

Learnable Up-sampling: Transpose Convolution

Recall: Typical 3 x 3 convolution, stride 1 pad 1



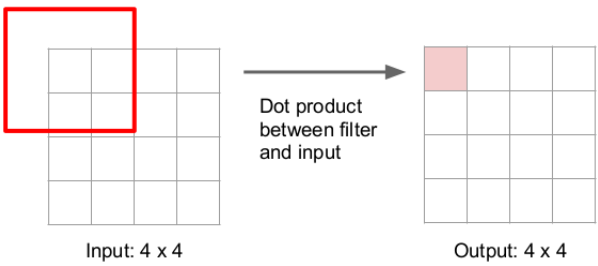
Input: 4 x 4



Output: 4 x 4

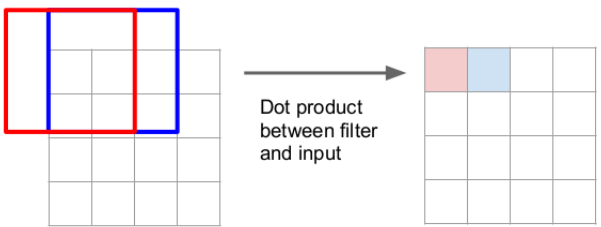
Learnable Up-sampling: Transpose Convolution

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



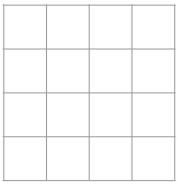
Learnable Up-sampling: Transpose Convolution

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



Learnable Up-sampling: Transpose Convolution

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

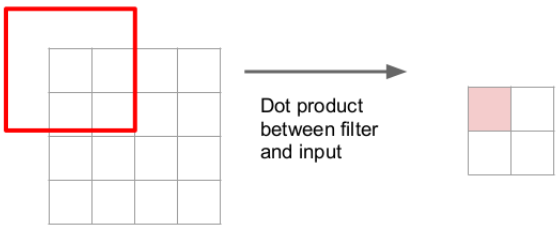


Input: 4 x 4

Output: 2 x 2

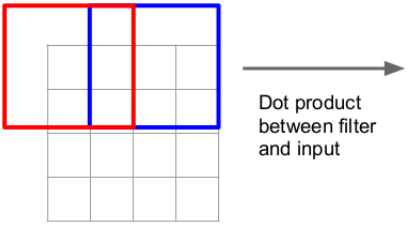
Learnable Up-sampling: Transpose Convolution

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



Learnable Up-sampling: Transpose Convolution

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



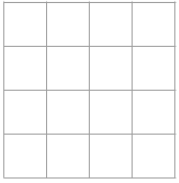
Filter moves 2 pixels in the input for every one pixel in the output

Stride gives ratio between movement in input and output

Learnable Up-sampling: Transpose Convolution

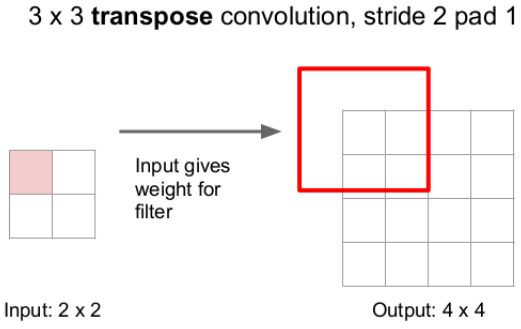
3 x 3 **transpose** convolution, stride 2 pad 1

Input: 2 x 2

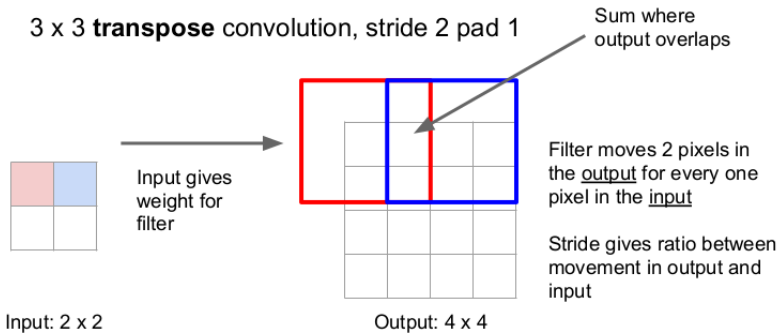


Output: 4 x 4

Learnable Up-sampling: Transpose Convolution

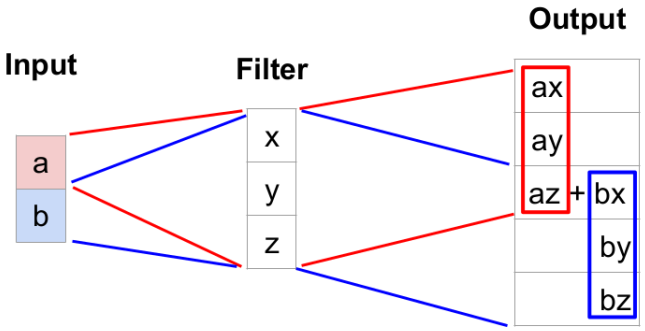


Learnable Up-sampling: Transpose Convolution



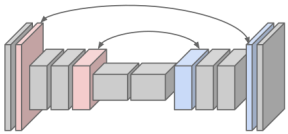
$\Longrightarrow$ Transpose convolution a.k.a. Deconvolution, Upconvolution, Fractionally strided convolution, Backward convolution, ..

Learnable Up-sampling: 1D Example



- Output contains copies of the filter weighted by the input, summing at where at overlaps in the output.
- Need to crop one pixel from output to make output exactly 2x input.

Semantic Segmentation Idea: Fully Convolutional



- Design network as a bunch of convolutional layers, with **down-sampling** and **up-sampling** inside[16].
- Down-sampling: Pooling, strided convolution $\Longrightarrow$ Up-sampling: Un-pooling, strided transpose convolution

[16] Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3431–3440.

Classification and Localization

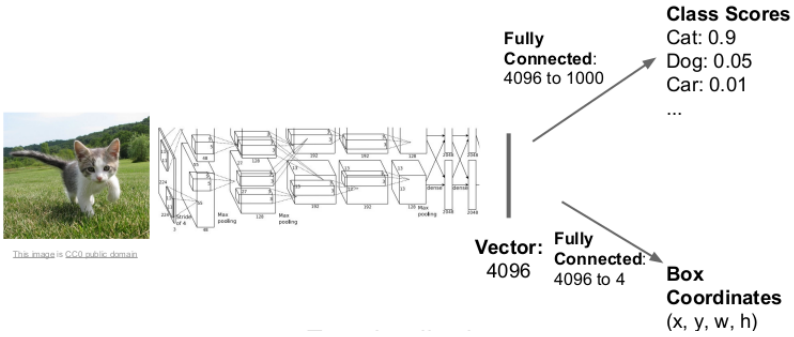


Figure 20: Single Object Detection

Classification and Localization

How would we train such a single object detection network that produces 4 + n values?

- Predicting the class of the object (n class probabilities) is a classification problem.
- Predicting the four coordinates for the bounding box is a regression problem.

$\Longrightarrow$ Need a loss function that combines these two problems.

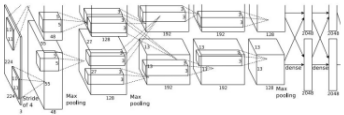
Classification and Localization

- Localization loss
  $\mathcal{L}_{loc} = \sum_{i=1}^{4} |x_{pred,i} - x_{truth,i}|$
- Confidence loss
  $\mathcal{L}_{conf} = \frac{-1}{N}(\sum_{i=1}^{N} y_{truth,i} \cdot log(y_{pred,i}))$

$\Longrightarrow$ Combined loss function

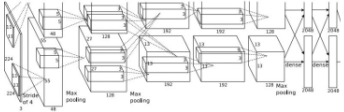
$$\mathcal{L} = \mathcal{L}_{loc} + \alpha \cdot \mathcal{L}_{conf}$$

Classification and Localization



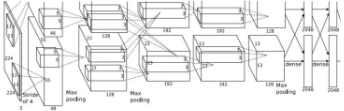
CAT: (x, y, w, h)

4 numbers



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

16 numbers



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
….

Many numbers!

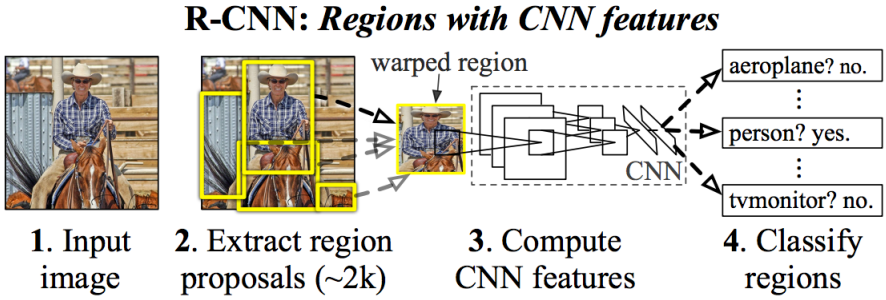


Figure 21: Region Convolutional Neural Network[17]

[17] Ross Girshick et al. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

Problems with R-CNN

- Extracting 2,000 regions for each image based on selective search.
- Extracting features using CNN for every image region. Suppose we have N images, then the number of CNN features will be N $\times$ 2,000.
- The entire process of object detection using R-CNN has three models:
  - CNN for feature extraction.
  - Linear SVM classifier for identifying objects.
  - Regression model for tightening the bounding boxes.

$\Longrightarrow$ Very slow, it takes around 40-50 seconds to make predictions for each image.

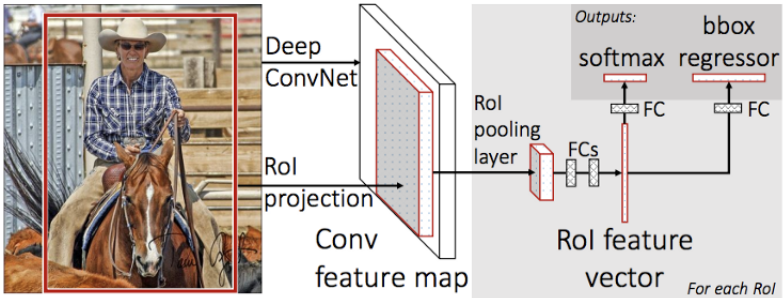
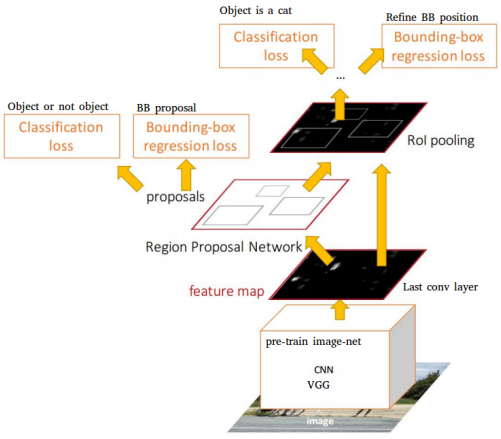


Figure 22: Fast R-CNN[18]

[18] Ross B. Girshick. "Fast R-CNN". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1440–1448.

$\Longrightarrow$ Still have a problem, it also uses selective search as a proposal method to find the Regions of Interest, which is a slow and time consuming process. It takes around 2 seconds per image to detect objects, which is much better compared to RCNN



Insert Region Proposal Network (RPN) to predict proposals from features. Jointly train with 4 losses:

- RPN classify object/not object
- RPN regress box coordinates
- Final classification score (object classes)
- Final box coordinates

Figure 23: Faster R-CNN

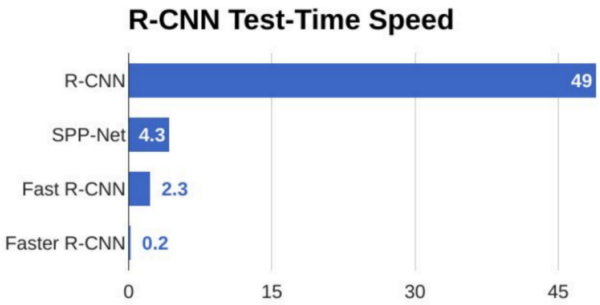


Figure 24: Experimentation results

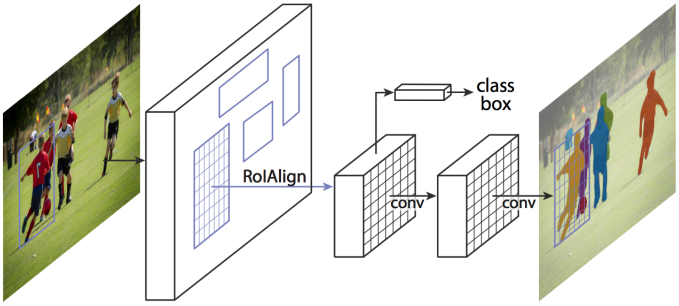


Figure 25: Mask R-CNN[19]

[19]Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.

Instance Segmentation

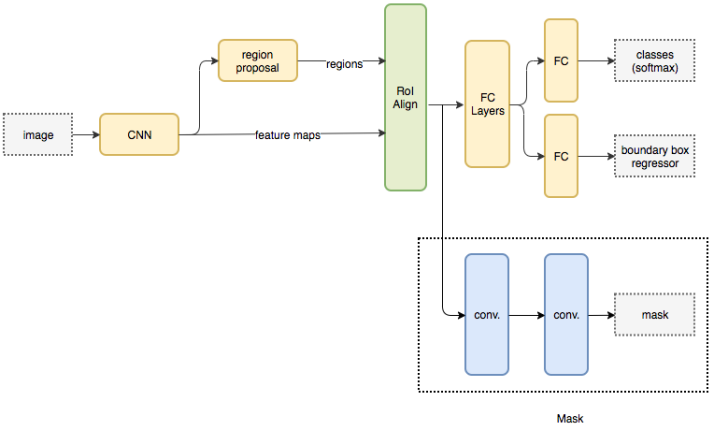


Figure 26: Mask R-CNN architecture

Figure 27: Amazing results

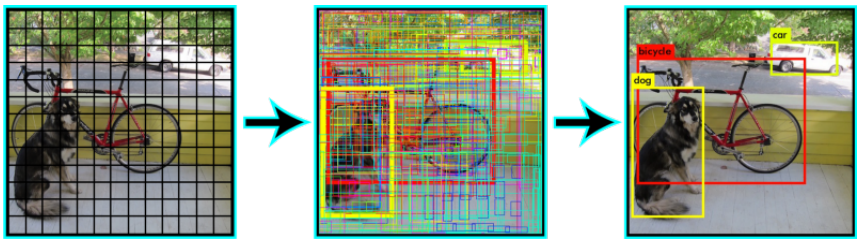
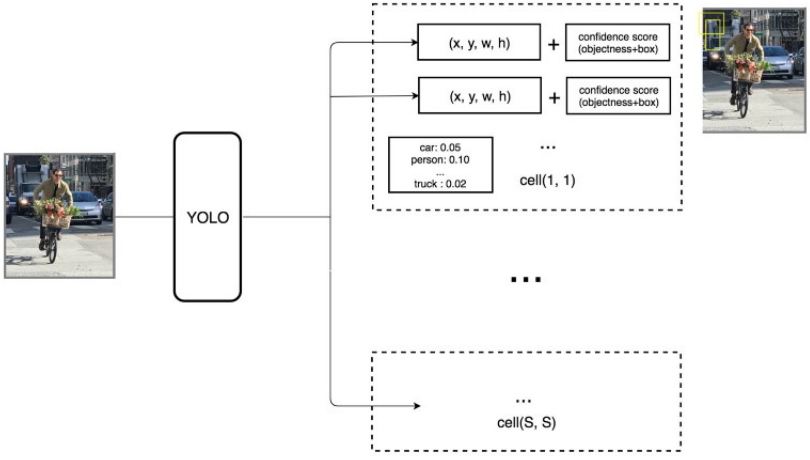


Figure 28: YOLO method motivation[20]

[20] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 779–788.

You Look Only Once



You Look Only Once

YOLO divide input image into $S \times S$ grid cells. It will predict:

- **B** boundary boxes and each box has one box confidence score (5 elements (x, y, w, h) and confidence score)
- **One** object only regardless of the number of boxes B
- **C** is a conditional class probabilities (one per class for the likeliness of the object class). The conditional class probability is the probability that the detected object belongs to a particular class.

$\Longrightarrow$ Output tensor shape $(S, S, B \times 5 + C)$

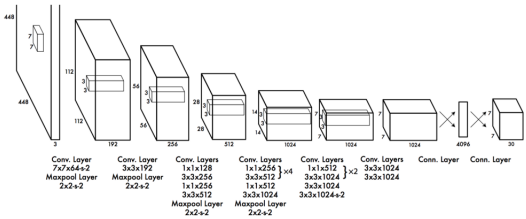


Figure 30: Network architecture design

YOLO has 24 convolutional layers followed by 2 fully connected layers (FC). Some convolution layers use 1×1 reduction layers alternatively to reduce the depth of the features maps.

YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of:

- Classification loss
- Localization loss
- Confidence loss

Final loss function:

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
\end{aligned}
$$

Intersection Over Union

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

Non-maxima Suppression

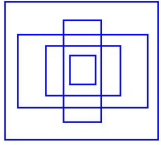
Non-maxima suppression algorithm

1. Sort the predictions by the confidence scores.
2. Start from the top scores, ignore any current prediction if we find any previous predictions that have the same class and IoU ¿ 0.5 with the current prediction.
3. Repeat step 2 until all predictions are checked.

YOLOv2: Accuracy Improvements

- Batch normalization: +2% mAP
- High-resolution classifier: +4% mAP
- Convolutional with Anchor Boxes
- Direct location prediction
- Fine-grained features
- Multi-scale trainning

YOLOv2: Convolutional with Anchor Boxes



Instead of predicting 5 arbitrary boundary boxes, we predict offsets to each of the anchor boxes above.

YOLOv3: An Incremental Improvement

- Class prediction: Replace softmax function with independent logistic classifiers (sum of output can be greater than 1 now). $\Longrightarrow$ Use binary cross-entropy loss for each label $\Longrightarrow$ Reduce computation complexity.
- Feature Pyramid Network (FPN)

YOLOv3: Performance

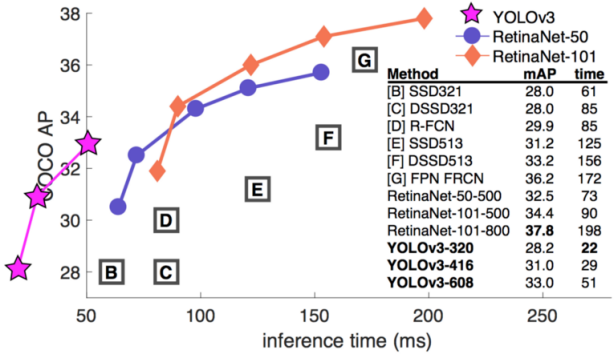
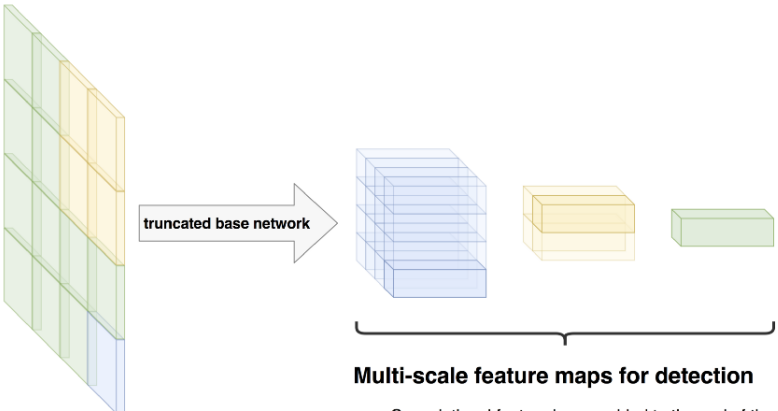


Figure 31: YOLOv3 Performance[21]

[21] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018).

Single Shot Multibox Detector (SSD)



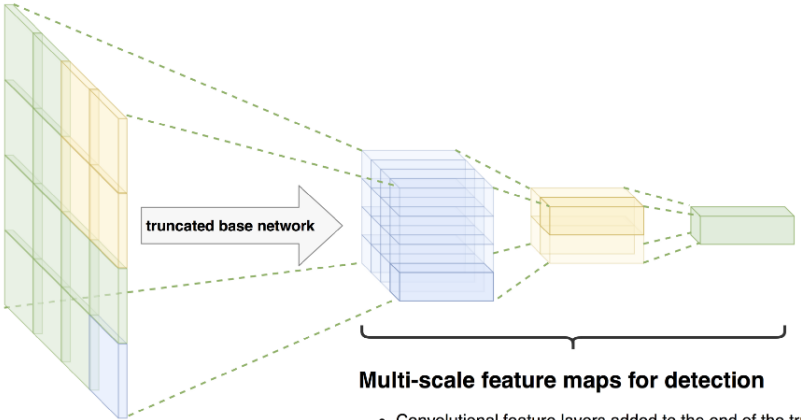
Input image

Shape i.e. [3, 224, 224]

Multi-scale feature maps for detection

- Convolutional feature layers added to the end of the truncated base network
- Decrease in size progressively and allow predictions of detections at multiple scales
- Shapes in this example: [256, 4, 4], [256, 2, 2], and [256, 1, 1]

Single Shot Multibox Detector (SSD)



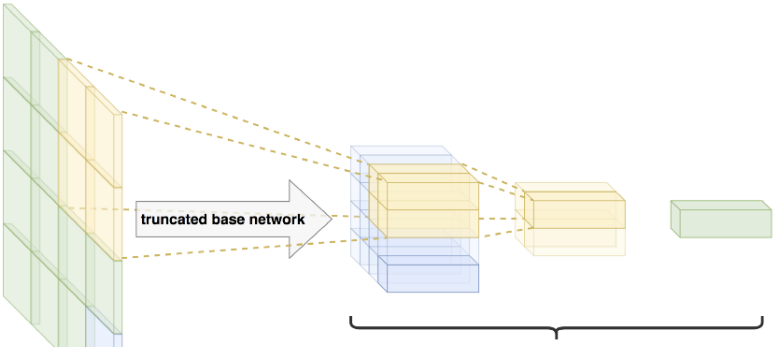
Input image

Shape i.e. [3, 224, 224]

Multi-scale feature maps for detection

- Convolutional feature layers added to the end of the truncated base network
- Decrease in size progressively and allow predictions of detections at multiple scales
- Shapes in this example: [256, 4, 4], [256, 2, 2], and [256, 1, 1]

Single Shot Multibox Detector (SSD)



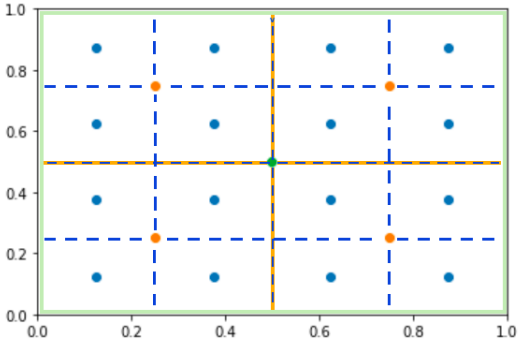
Input image

Shape i.e. [3, 224, 224]

Multi-scale feature maps for detection

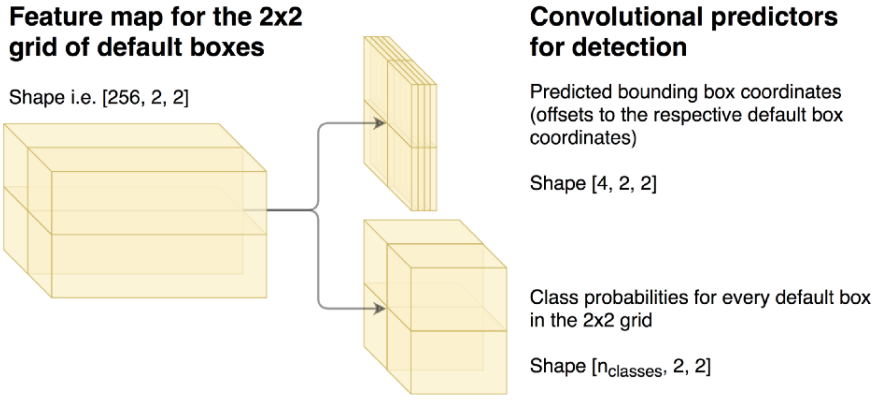
- Convolutional feature layers added to the end of the truncated base network
- Decrease in size progressively and allow predictions of detections at multiple scales
- Shapes in this example: [256, 4, 4], [256, 2, 2], and [256, 1, 1]

Single Shot Multibox Detector (SSD)



- Green dots: 4×4 boxes
- Orange dots: 2×2 boxes
- Blue dots: 1×1 boxes

Single Shot Multibox Detecto (SSD)



Fed every one of the three feature maps is input to two more convolutional layers.

Trade off between speed and accuracy

Let's summarize something:

- We defined several grids of differently sized default boxes that will allow us to detect objects at different scales in one single forward pass. Many previous architectures detected objects at different scales for example by passing differently sized versions of the same picture to the detector which is computationally more expensive.
- For each default box in every grid, the network outputs n class probabilities and 4 offsets to the respective default box coordinates which give the predicted bounding box coordinates.

$\Longrightarrow$ But still less accurate than two-stage methods.

Trade off between Speed and Accuracy

- The advantage of two-stage methods try to show off a **few region of interest** - ignore background.
- Meanwhile single-stage methods (like YOLO, SSD) evaluate roughly between ten to hundred thousand candidate locations (way more than the 4x4, 2x2 + 1 default boxes in our example here) and of course most of these boxes do not contain an object.

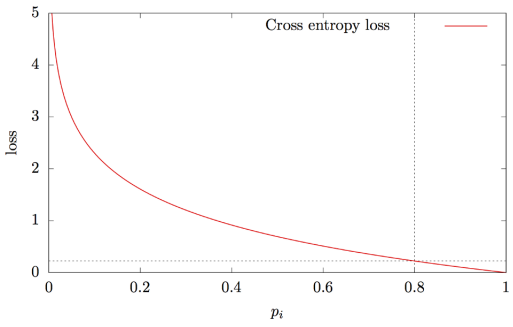
  $\Longrightarrow$ Imbalance class

Trade off between Speed and Accuracy

- Cross-entropy loss function, where i is the index of class, y_i is the label (1 or 0), and p_i is the predicted propability that object belongs to class i.

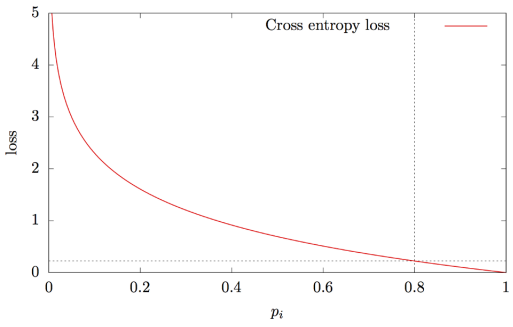
$$C(p, y) = -\sum_i y_i \log_{p_i}$$

- Lets say a box contains background and the network is 80% sure that it actually is only background. In this case $y_{background} = 1$, all other y_i are 0 and $p_{background} = 0.8$.



Trade off between Speed and Accuracy

- We can see that at 80% certainty that the box contains only background, the loss is still ~ 0.22.
- Imagine: We have ten actual objects in the image and the network is not really sure to which class they belong so that their loss is i.e. ~ 3. This would give us around 30

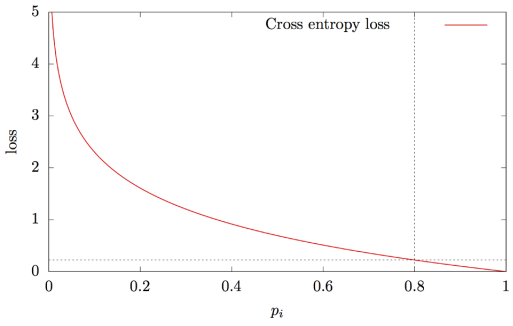


Trade off between Speed and Accuracy

- All the another $\sim$ 10000 default boxes are background and the network is 80% sure that they are just background.

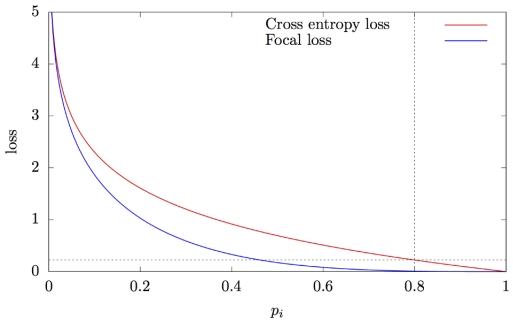
  $\Longrightarrow$ This give us a loss of around:

  $20 \times 0.22 = 2200$



Focal Loss[22]

- Lin et al. (2017) propose a loss function so that the learning can focus on the few interesting cases. The authors called their loss function **Focal loss** and their architecture **RetinaNet**.
  $C(p, y) = -\sum_i y_i (1 - p_i)^\gamma \log_{p_i}$



[22] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

Focal Loss

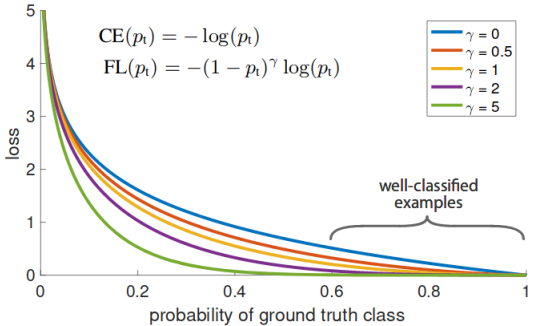


Figure 32: Focal loss with some γ factor[23]

[23] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *IEEE transactions on pattern analysis and machine intelligence* (2018).

Remark

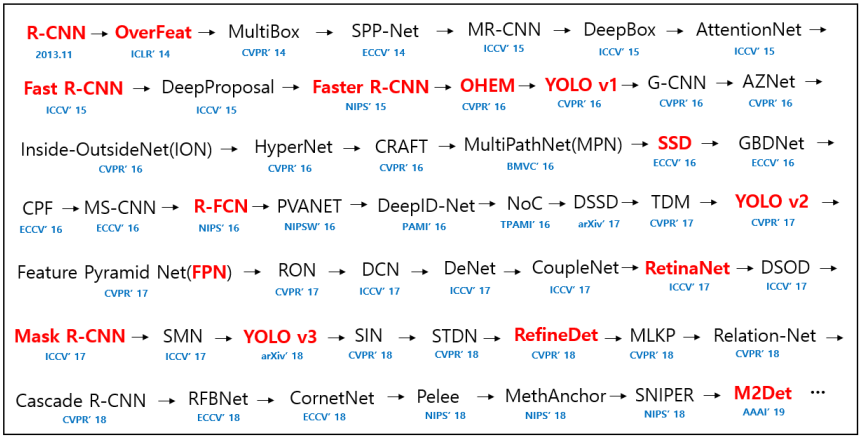


Figure 33: Deep Learning methods for Object detection history[Lee Hoseong 2019]

Q&A

References

- Le Cun et al. “Handwritten Digit Recognition with a Back-Propagation Network”. In: *Advances in Neural Information Processing Systems*. Morgan Kaufmann, 1990, pp. 396–404.
- Ross B. Girshick. “Fast R-CNN”. In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1440–1448.
- Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

References ii

- Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.
- Kaiming He et al. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 2980–2988.
- Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in Neural Information Processing Systems*. P. 2012.
- Yann Lecun et al. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE*. 1998, pp. 2278–2324.
- Min Lin, Qiang Chen, and Shuicheng Yan. *Network in Network*. 2013.

- Tsung-Yi Lin et al. “Focal Loss for Dense Object Detection”. In: *IEEE transactions on pattern analysis and machine intelligence* (2018).
- Aäron van den Oord et al. “WaveNet: A Generative Model for Raw Audio”. In: *SSW*. 2016.
- Joseph Redmon and Ali Farhadi. “YOLOv3: An Incremental Improvement”. In: *CoRR* abs/1804.02767 (2018).
- Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 779–788.

- Evan Shelhamer, Jonathan Long, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 3431–3440.
- Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition*. 2014.
- Christian Szegedy et al. *Going deeper with convolutions*. 2014.
- Ashish Vaswani et al. "Attention Is All You Need". In: *NIPS*. 2017.
- Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016).