Intelligent Systems: Reasoning and Recognition

James L. Crowley

Ensimag 2 — Winter Semester 2021
Lesson 12 — 19 March 2021

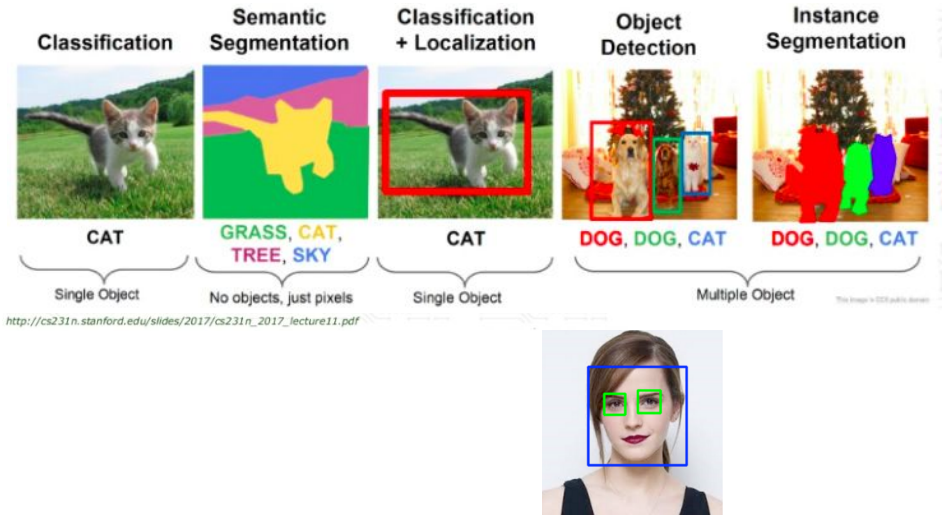
Locating Patterns in Images

Outline

Introduction

Computer Vision Tasks used in ML challenges

Much of the early work on Machine Learning for Vision was focused on classification of pixels and images. The following is a taxonomy of such problems:



Hierarchical Part Detection

1) Image Classification: Does an image (or imagette) contain an instance of a class?
2) Semantic Segmentation: What is the most likely class for each pixel?
3) Single Object Detection: Does the image contain an instances of a class, and if so, at what position and scale?
4) Multiple Object Detection: Does the image contain instances of several classes, and if yes, what are the positions and scales for each instance?
5) Hierarchical part detection. Where are the component parts for an object class.

Many (or most) of the widely used Network Architectures were designed specifically to compete in computer vision challenges, in which competitors compete on formalized vision tasks using publicly available data sets. The nature of these tasks and particularly the specification for the Benchmark data sets helps explain many of the parameters of the architecture designs.

Benchmark Data Sets Visual Task Challenges

As we saw in lesson 11, many of the popular architectures were designed specifically to address research challenges based on image data sets. Classically, these data sets were for challenges related to object detection. More recently the challenges increasingly address other visual tasks.

The ImageNet Challenge for Object Detection

ImageNet was originally concerned with Image Classification: Does an image (or imagette) contain an instance of a class? Most state-of-the-art object detection networks pre-train on ImageNet and then rely on transfer learning to adapt the learned recognition system to a specific domain. The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) uses a "trimmed" list of only 1000 image categories or "classes", including 90 of the 120 dog breeds classified by the full ImageNet schema.

ImageNet crowdsources its annotation process. In 2018 there were more than 14 million images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided. Image-layer annotations indicate the presence or absence of an object class in an image. Object-layer annotations provide a bounding box around the (visible part of the) indicated object.

In 2014, more than fifty institutions participated in the ILSVRC, almost exclusively with different forms of Network Architectures. In 2017, 29 of 38 competing teams in the ILSVRC demonstrated error rates less than 5% (better than 95% accuracy).

However, the ILVSRC task is to identify images as belonging to one of a thousand categories; humans can recognize a larger number of categories, and also (unlike the programs) can judge the context of an image. More importantly, humans are capable of MANY other visual tasks involving Spatio-temporal interaction with 3D. In cognitive psychology, these are referred to as visual competences.

COCO - Common Objects in Context

Microsoft COCO is a large-scale object detection, segmentation, and captioning dataset created in 2015. Images in the COCO data set display are everyday objects captured from everyday scenes. This adds some “context” to the objects captured in the scenes

COCO contains more than 2.5M instances in 91 object categories, with 5 captions per image 330K images (200K+ annotated) with 250,000 people with key points.

Data sets for other visual tasks

An extensive (very large) list of publically available benchmark data sets and research challenges for visual tasks may be found at.
https://en.wikipedia.org/wiki/List_of_datasets_for_machine-learning_research
This list continues to grow rapidly.

There is a growing interest in developing techniques for recognizing actions and activities from video sequences and multimodal data. The recent emergence of generative techniques, combined with rapid advances in Robotics and Autonomous Systems appear likely greatly expand this set of tasks. In particular the recent progress in Transformers and Attention-based techniques in Natural Language processing appear likely to enable many new competences for computer vision.

The following are some techniques for multiple object detection and semantic detection.

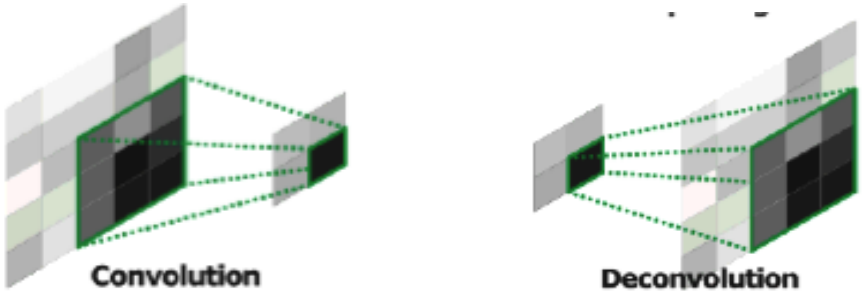
Generative Convolutional Networks

Generating images with deconvolution.

Just as it is possible to generate signals from codes using fully connected generative networks, it is possible to construct Generative Convolutional Networks for CNNs using an operation known as deconvolution.

Deconvolution is often used with convolutional networks to determine the location of a detected pattern in an image. Deconvolution provides a coarse pixel-wise label map that segments the image into regions corresponding to recognized classes and can be used for semantic segmentation.

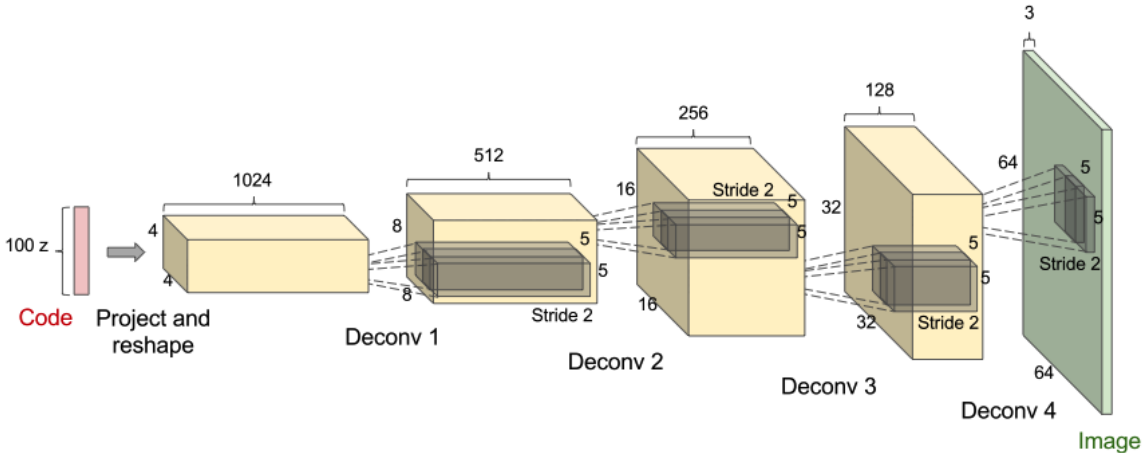
De-convolution treats the learned receptive fields as basis functions, and uses the activation at level *l* to create a weighted sum of bases at level *l+1*. The learned receptive fields are multiplied by the map of activation at level *l* to generate overlapping projections of receptive fields. These are then summed to create an image at level *l+1*. In some cases, the boundary is cropped to obtain an image at the target window size.



A stride greater than 1 can be used to create a larger image. The stride acts as the opposite of pooling. For 2x2 average pooling, de-convolution simply projects 4 displaced copies of the receptive field onto a 2 x 2 grid of overlapping receptive fields. These are then summed to give an image. An example of such a network is DCGAN architecture.

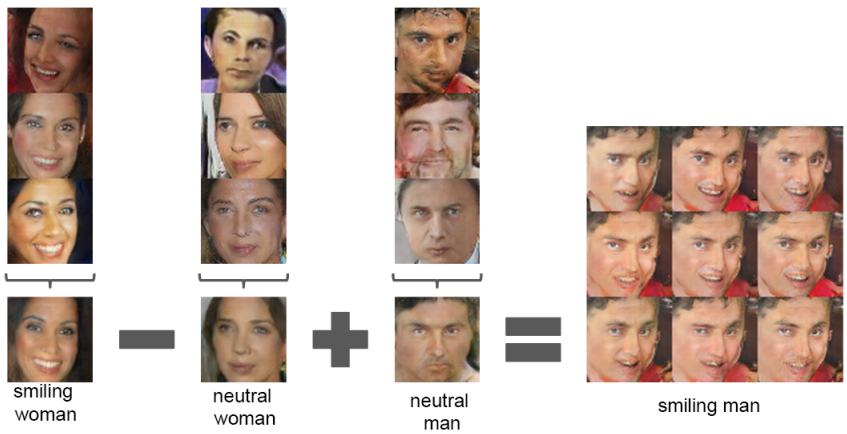
DCGAN

A DCGAN (deep convolutional generative adversarial network) takes 100 random numbers as an input (or code) and outputs an color image of size 64x64x3



The first fully connected layer is a 4 x 4 array of 1024 cells (Depth = 1024). Total number of cells is 16 K. This layer has 160 K weights and 16 K biases to train. This first layer is deconvolved into an 8 x 8 by 512 second layer, where deconvolution projects each of the cells in the 4x4 layer onto an overlapping set of 5x5 receptive field with a stride of 2. The process is repeated to create a 3rd layer that is 16x16x256 and then a 4th layer that is 32 x 32 by 128. The final output is a 5th layer with 64 x64 pixels of 3 colors.

The following are some examples of images generated using DCGAN:

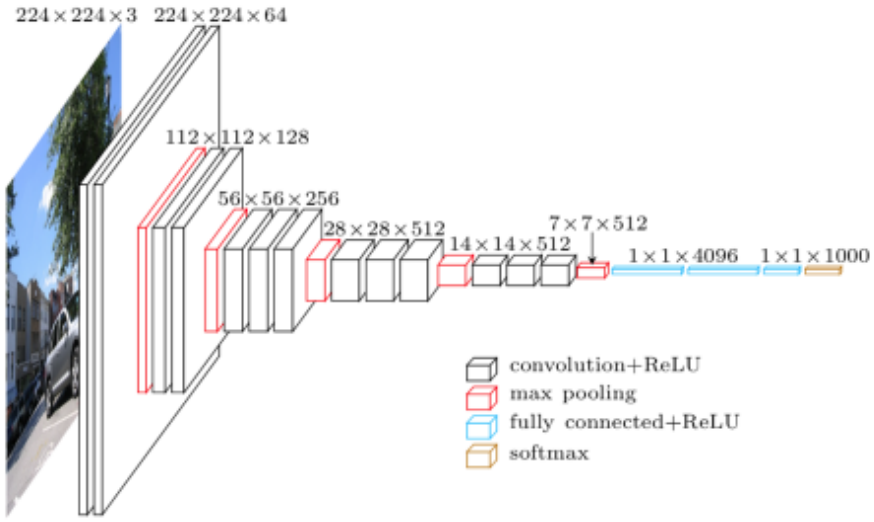


Example smiling man images generated from smiling woman images.

From:

Radford, A., Metz, L., and Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks, ICLR 2016.

Deconvolution with VGG16



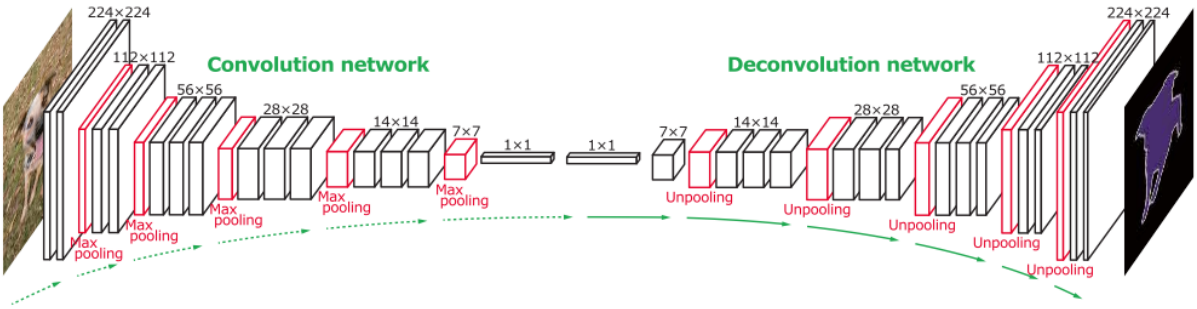
VGG16 is a convolutional neural network architecture proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". VGG16 scored 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes.

VGG16 improves on AlexNet by replacing large kernel-sized filters (11 x 11 and 5 x 5) with a cascade of 3×3 kernel-sized filter. VGG16 was trained for weeks and using NVIDIA Titan Black GPU's.

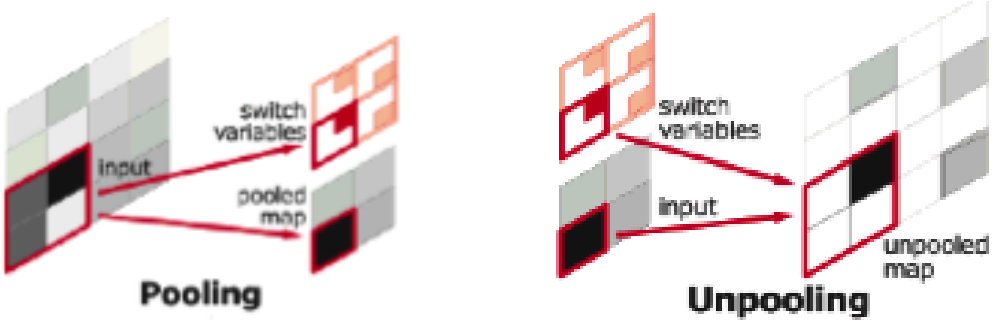
VGG accepts a 224 x 224 RGB image as input. The first 17 layers use 3x3 convolutions, relu and 2x2 max pooling with a stride of 2 after layers 2, 4, 7, 10 and 13. The depths are D=64 (layers 1, 2), D=128 (layers 3, 4), D=256 (layers 5, 6, 7). D=512 (layers 8 to 13). Layers 14 and 15 are a 1 x 1 convolution with depth 4096. Layer 16 is 1 x 1 x 1000 likelihood score for 1000 pretrained classes using softmax activation.

Three Fully-Connected (FC) layers follow a stack of 1x1 convolutional layers. The first two full-connected layers have 4096 channels each. The third layer has 1000 channels corresponding to the 1000 image classes corresponding to the 1000 image-net classes used in the ILSVRC (Image-net Large Scale Visual Recognition Classification) challenge for which it was designed. The final layer uses soft-max activation to determine the most likely classes in the 224 x 224 input image.

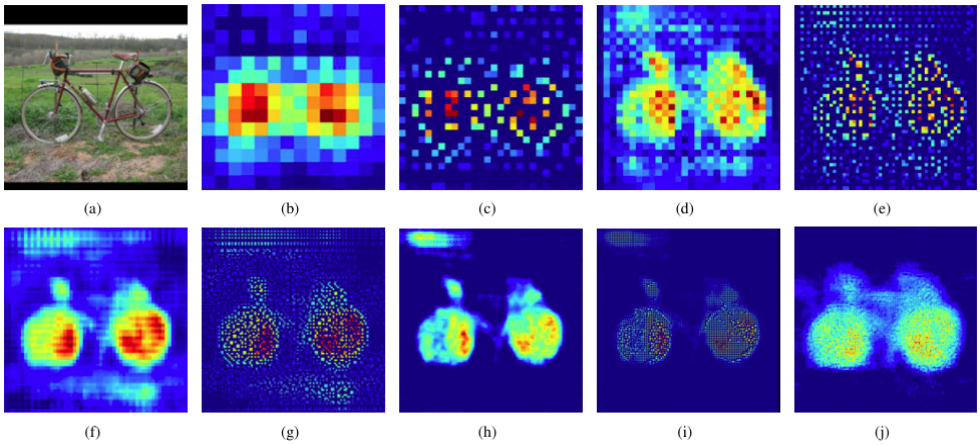
Normally VGG16 is used by scaling (texture mapping) the input image into a 224 by 224 window, without regard for the scale of the input, and produces only a probability for 1000 trained classes in the image. However, VGG16 can be adapted as a multiple object detector using deconvolution. The deconvolution network is a mirror image, replacing pooling with "un-pooling" and convolution with "deconvolution". This is often referred to as a U-net encoder-decoder.



VGG uses max pooling. With Max pooling, unpooling requires remembering which unit was selected for each pooling operation. This is done with a "switch Variable" that records the selected unit. The output is a larger sparse layer in which 3/4 of the activations are zero.

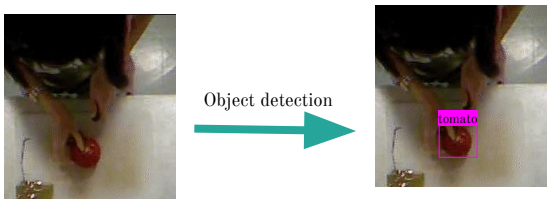


The following shows an example with deconvolution of the VGG net of a bicycle. (a) is the original image. The other images show the results of max-pooling for the 14x14, 28x28, 56x56, 112x112, and 224x224 layers

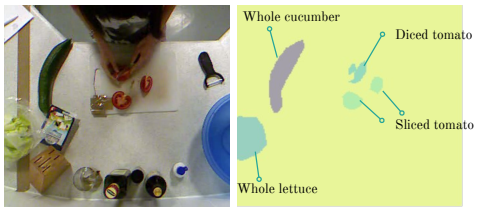


The output pixels can be used to provide scores for semantic segmentation for each pixel. Alternatively bounding boxes can be estimated by computing the 1st and 2nd moments (center of gravity and covariance), with a likelihood provided by the zeroth moment (sum of pixel class likelihoods) for each class.

For example, the following are multi-class object detection and semantic segmentation images obtained from deconvolution with VGG taken from Nachwa Aboubakr's thesis on observation of cooking activities. Her experiments use the 50 Salads data set.



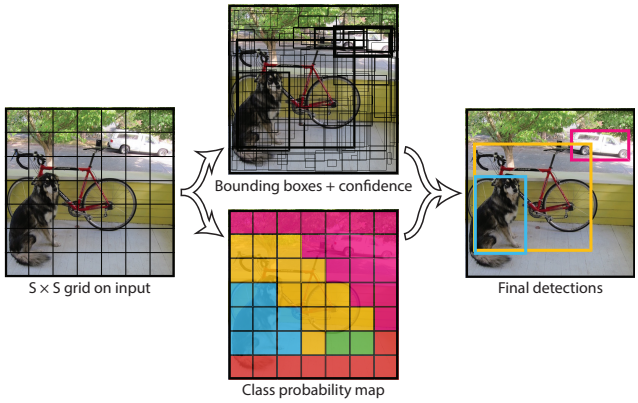
Object **label** + object **location**



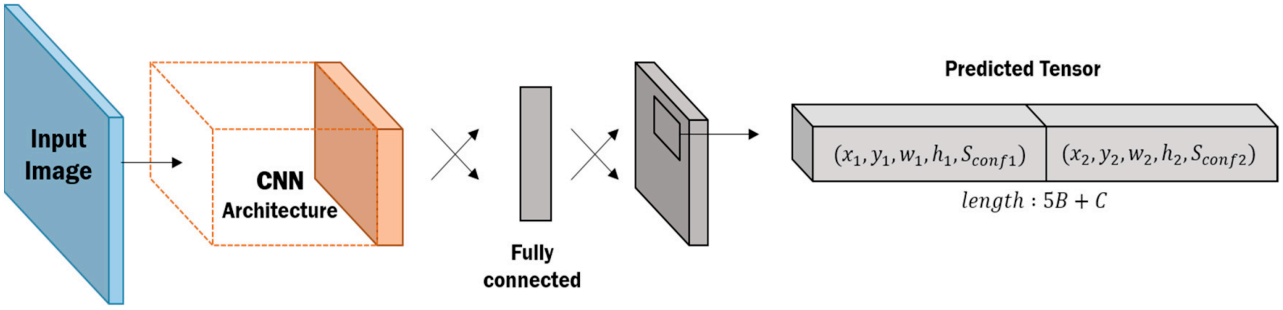
Example of pixel-wise labeling of a scene

YOLO: You Only Look Once

YOLO poses object detection as a single regression problem that estimates bounding box coordinates and class probabilities at the same time directly from image pixels. A single convolutional network simultaneously predicts multiple bounding boxes and class probabilities for each box in a single evaluation. The result is a unified architecture for detection and classification that is very fast.



The input image is divided into an S x S grid of cells. Each grid cell predicts B bounding boxes as well as C class probabilities. The bounding box prediction has 5 components: (x, y, w, h, confidence).



(From Kim, J. and Cho, J. Exploring a Multimodal Mixture-Of-YOLOs Framework for Advanced Real-Time Object Detection. Applied Sciences, 2020, vol. 10, no 2, p. 612.)

The (x, y) coordinates represent the center of the predicted bounding box, relative to the grid cell location. Width and height (w, h) are predicted relative to the entire image.

Both the (x, y) coordinates and the window size (w, h) are normalized to a range of [0,1]. Predictions for bounding boxes centered outside the range [0,1] are ignored. If the predicted object center (x, y) coordinates are not within the grid cell, then object is ignored by that cell.

Each grid cell also predicts C class conditional probabilities $P(Class_i \mid Object)$

These are conditioned on the grid cell containing an object. Only one set of class probabilities are predicted per grid cell, regardless of the number of boxes.

Confidence is computed as $CF = P(Class_i)IoU(predicted,True)$

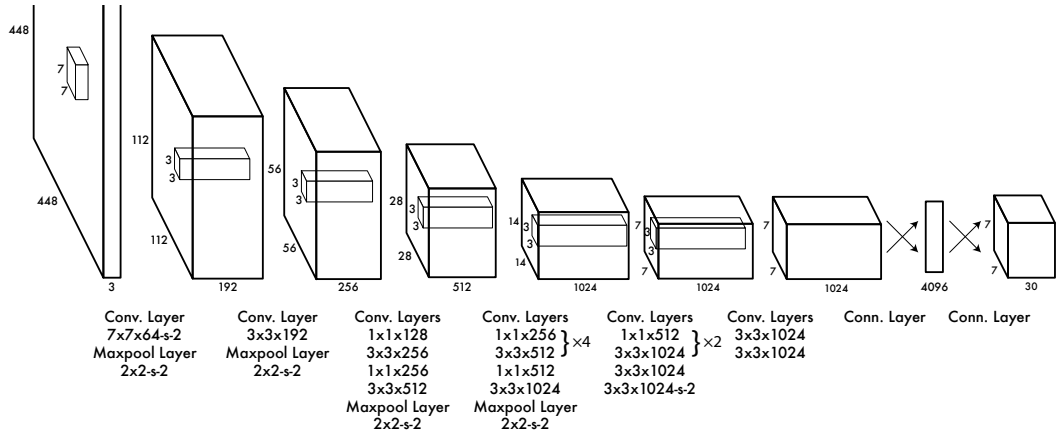
This can be evaluated using Bayes rule:

$$P(Class_i)IoU(predicted,True) = P(Class_i \mid Object)P(Object)IoU(predicted,True)$$

These predictions are encoded as an S x S x (5B+C) tensor. Where SxS is the number of grid cells, B is the number of Bounding Boxes predicted and C is the number of image classes. For the Pascal visual Object Classification challenge, S = 7, B = 2 and C=20 yielding a 7x7x30 tensor.

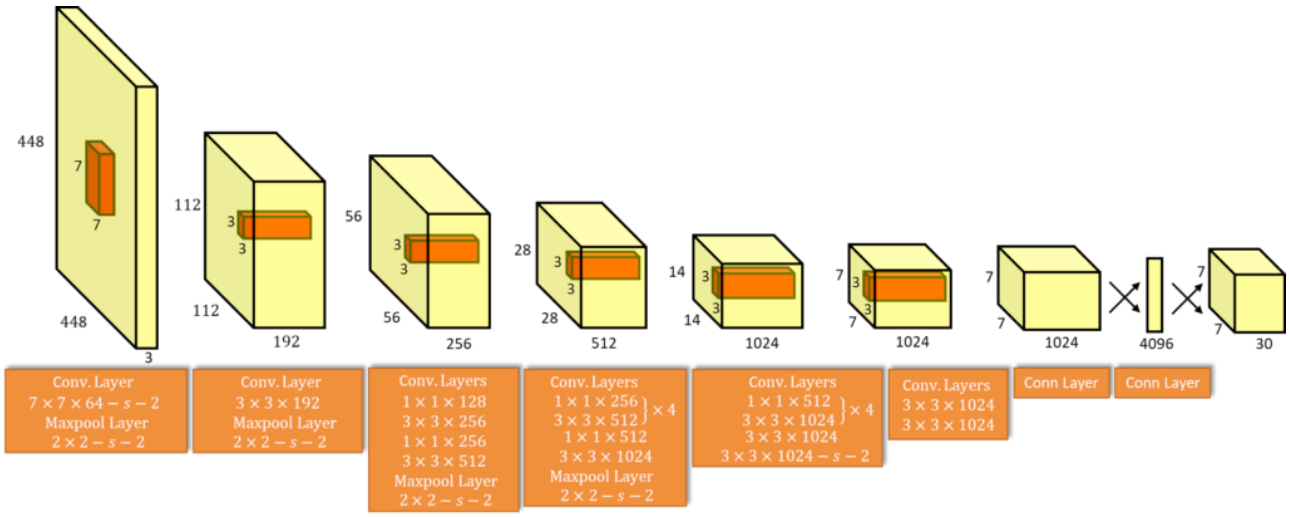
These scores encode the probability of a member of class *i* appearing in a box, and how well the box fits the object. If no object exists in a cell, the confidence score should be zero. Otherwise the confidence score should equal the intersection over union (IOU) between the predicted box and the ground truth.

The Yolo-1 Network



(From: REDMON, J., DIVVALA, S., GIRSHICK, R., et al. You only look once: Unified, real-time object detection. In : Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 779-788.)

The network was inspired by GoogleLeNet. The detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1 by 1 convolutional layers reduce the features space from preceding layers.



(from: http://datahacker.rs/how-to-peform-yolo-object-detection-using-keras/)

The convolutional layers were pretrained on the ImageNet data-set at half the resolution (224 by 224 input image). Image resolution was then doubled to (448 x 448) for detection.

Layer Name	Filters	Stride	Output Dimension
Conv 1	7 x 7 x 64	2	224 x 224 x 64
Max Pool 1	2 x 2	2	112 x 112 x 64
Conv 2	3 x 3 x 192	1	112 x 112 x 192
Max Pool 2	2 x 2	2	56 x 56 x 192
Conv 3	1 x 1 x 128	1	56 x 56 x 128
Conv 4	3 x 3 x 256	1	56 x 56 x 256
Conv 5	1 x 1 x 256	1	56 x 56 x 256
Conv 6	1 x 1 x 512	1	56 x 56 x 512
Max Pool 3	2 x 2	2	28 x 28 x 512
Conv 7	1 x 1 x 256	1	28 x 28 x 256
Conv 8	3 x 3 x 512	1	28 x 28 x 512
Conv 9	1 x 1 x 256	1	28 x 28 x 256
Conv 10	3 x 3 x 512	1	28 x 28 x 512
Conv 11	1 x 1 x 256	1	28 x 28 x 256
Conv 12	3 x 3 x 512	1	28 x 28 x 512
Conv 13	1 x 1 x 256	1	28 x 28 x 256
Conv 14	3 x 3 x 512	1	28 x 28 x 512
Conv 15	1 x 1 x 512	1	28 x 28 x 512
Conv 16	3 x 3 x 1024	1	28 x 28 x 1024
Max Pool 4	2 x 2	2	14 x 14 x 1024
Conv 17	1 x 1 x 512	1	14 x 14 x 512

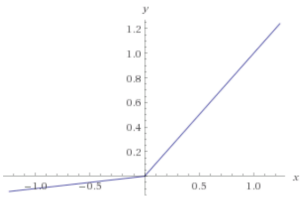
Conv 18	3 x 3 x 1024	1	14 x 14 x 1024
Conv 19	1 x 1 x 512	1	14 x 14 x 512
Conv 20	3 x 3 x 1024	1	14 x 14 x 1024
Conv 21	3 x 3 x 1024	1	14 x 14 x 1024
Conv 22	3 x 3 x 1024	2	7 x 7 x 1024
Conv 23	3 x 3 x 1024	1	7 x 7 x 1024
Conv 24	3 x 3 x 1024	1	7 x 7 x 1024
Fully-Connected 1	-	-	4096
Fully-Connected 2	-	-	7 x 7 x 30 (1470)

Training YOLO

The 20 convolutional layers followed by average pooling and a fully connected layer were first pre-trained using the ImageNet 1000-class dataset images transformed to a resolution of 224x224. Training for the full network then continued using the same images rescaled to 448 by 448, with data augmentation using random scaling and translations, and randomly adjusting exposure and saturation. The full network was trained for 135 epochs using a batch size of 64, momentum of 0.9 and decay of 0.0005. Training for the first 75 epochs used a learning rate of 0.01. After 75 epochs the learning rate was slowly lowered to 0.001

All layers except the final layer use a leaky rectified linear activation function:

$$f(z) = \begin{cases} x & \text{if } x > 0 \\ 0.1x & \text{otherwise} \end{cases}$$



The final layer used simple linear activation.

Loss Function for YOLO

The YOLO loss function is based on sum of squared errors. However, to avoid instability in training due to simultaneous estimation of bounding boxes and classes, errors in estimation are bounding boxes and classes weighted differently, with the two terms λ_{coord}= 5 and λ_{noobj}=0.5. Rather than estimate w and h directly, the network estimates the square root of w x h. The resulting loss function is

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

where 1_i^{obj} is 1 if an object appears in cell i, and 1_{ij}^{obj} is 1 if the j^{th} bounding box predictor in cell *i* is responsible for the prediction of the object. This loss function only penalizes classification error if an object is present in that grid cell. It also only penalizes the bounding box coordinate error if that predictor is "responsible" for the ground truth box.

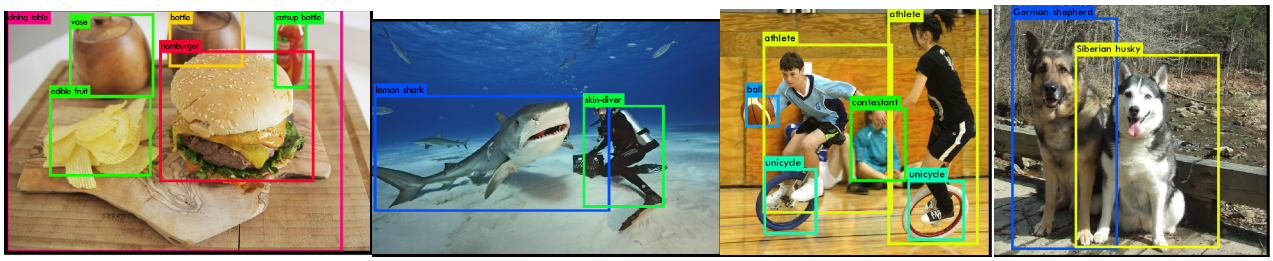
Training uses dropout with a rate of 0.5 after the first connected layer to avoid over-fitting. The training data is augmented using random scaling and translations of up to 20%. Saturation and exposure of the color vector are randomly adjusted by up to 1.5 using HSV color-coding.

Limitations of YOLO-1

The first generation of YOLO imposed strong spatial constraints on bounding box predictions, because each grid cell only predicts two boxes and can only have one class. This spatial constraint limits the number of nearby objects that our model can predict. Thus the network struggles with small objects that appear in groups, such as flocks of birds.

The network has difficulty recognizing objects in new or unusual aspect ratios or configurations. Our model also uses relatively coarse features for predicting bounding boxes since our architecture has multiple down-sampling layers from the input image. Also, the loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a large box is generally benign but a small error in a small box has a much greater effect on IOU. The main source of error is incorrect localizations.

YOLO-9000 (YOLOv2)



In 2016, the YOLO team published performance evaluation results and source code for a new version of YOLO referred to as Yolo-9000. Yolo-9000 employed a number of innovations, including ideas that had emerged in the machine learning literature the previous year. These included:

Batch Normalization

Batch normalization provided a significant improvement in convergence while eliminating the need for other forms of regularization. Batch normalization provided a 2% improvement in mAP (mean average precision), and made it possible to remove dropout from the architecture without overfitting.

Higher Resolution Classifier

The original YOLO was first trained with 224x224 images and then with 448x448 images for bounding box detection. For Yolo-2, the classification network was trained at 448x448 resolution for 10 epochs on ImageNet. Training with higher resolution network provided an increase of almost 4% mAP.

Convolutional With Anchor Boxes.

YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. For Yolo-2, the authors removed the fully connected layers from YOLO and used anchor boxes to predict bounding boxes. Input images were reduced to 416x416 providing an output feature map of 13x13. The class prediction mechanism is decoupled from spatial location, by predicting the objectness for each anchor box.

Dimension Clusters.

Anchor box dimensions are initially hand picked. The authors replaced this with k-means clustering on the training set bounding boxes to automatically find good priors. The resulting cluster centroids are significantly different than hand-picked anchor boxes, with fewer short, wide boxes and more tall, thin boxes.

Bounding boxes with dimension priors and location prediction.
Yolo-9000 predicts the width and height of the box as offsets from cluster centroids. The center coordinates of the box are predicted relative to the location of filter application using a sigmoid function.

Fine-Grained Features
Yolo-9000 predicts detections on a 13x13 feature map. While this is sufficient for large objects, it may benefit from finer grained features for localizing smaller objects. To correct this, Yolo-9000 includes a pass-through layer that brings features from an earlier layer at 26x26 resolution.

Multi-Scale Training
Because the network uses only convolutional and pooling layers it can be resized on the fly. To obtain robustness to scale, the network was trained with images of different sizes.

During training, the input image size was changed every few iterations. Every 10 batches our network randomly adopted a new image resolution. Because model down-samples by a factor of 32, the resolutions are multiples of 32 from 320x320 to 608x608. This forces the network to learn to predict well across a variety of input dimensions. Thus same network can predict detections at different resolutions. The network runs faster at smaller sizes giving an easy tradeoff between speed and location accuracy.

At low resolutions YOLOv2 operates as a cheap, fairly accurate detector. At 288x288 it runs at more than 90 FPS. This makes it ideal for smaller GPUs, high framerate video, or multiple video streams.

At high resolution the network is competitive with the state of the art giving 78.6 mAP on VOC 2007 while still operating above real-time speeds

Code and pre-trained models for Yolo-9000 are available on-line at http://pjreddie.com/yolo9000/.

Additional incremental improvements have been provided for YOLOv3 and YOLOv4.