Convolutional Neural Networks

James L. Crowley[1]

[1] Grenoble Insitut Polytechnique, Univ. Grenoble Alpes
http://crowley-coutaz.fr/jlc/jlc.html

Abstract. This chapter presents Convolutional Neural Networks (CNNs). The chapter begins with a review of the convolution equation, and a description of the original LeNet series of CNN architectures. It then traces the emergence of Convolutional Networks as a key enabling technology for Computer Vision resulting from the publication of AlexNet at the 2012 ImageNet Large Scale Image Recognition Challenge. This is followed by a description of the VGG architecture and the YOLO Single Shot Detection network for Image Object Detection.

Learning Objectives: This chapter presents Convolutional Neural Networks, with a summary of the history, fundamental theory, and a review of popular architectures that have played a key role in the emergence of Deep Learning as an enabling technology for Artificial Intelligence. After reading this chapter, students will be able to understand the basic principles of convolutional neural networks and how such networks can be used to detect patterns in signals. Students will understand the meaning and significance of network hyper-parameters, and be able to select among the commonly used architectures such as VGG and YOLO to solve problems in pattern analysis and signal detection in audio, visual and other forms of multidimensional signals.

Keywords: Convolutional Neural Networks (CNNs), Hyper-parameters, CNN Architectures, LeNet, AlexNet, VGG, You Only Look Once (YOLO).

1. Convolutional Neural Networks.

During the second wave of popularity of Neural Networks in the 1980s, researchers began experimenting with networks for computer vision and speech recognition. Direct application of neural networks in these domains required training networks with an excessively large number of parameters, greatly exceeding the memory and computing power of available computers. For example, direct recognition of the 44 English phonemes (speech elements) in a speech signal required a network capable of processing an audio signal composed of *1600* samples. A fully connected two layer network with 1600 hidden units in the first layer and 44 output units in the second layer would have more than *2.5* Million trainable parameters, while typical computer memory address spaces in this period were less than 1 Million bytes. In the case of computer vision, the situation was even more extreme. A typical digital image at that time was sampled at 512 x 512 rows and columns, represented by 2^{18} (or 256 K) 8-

bit grayscale pixels. Training a fully connected 2-layer perceptron to recognize a large set of objects in an image was not a serious proposition.

Inspiration for a solution was provided by neuroscience. In the early 1960s, David Hubel and Torsten Wiesel [1] fixed a cat's head in a rig and probed the visual cortex with electrodes while scanning patterns of light on a screen, as shown in figure 1.They found that individual cells in the visual cortex responded to specific patterns of light at specific locations and sized. They referred to these patterns as receptive fields. By systematic probing, they found that the visual cortex of the cat is composed of layers of retinotopic maps that respond to patterns of spots, bars, and edges at a narrow range of positions, sizes and orientations. Subsequent research showed that the receptive fields could be modeled as local filters for spatial frequency patterns at different spatial frequency bands and orientation. As they moved through the visual cortex, Hubel and Weisel found that these patterns were combined to form more complex patterns, such as corners and crosses. These more complex patterns were named "complex" receptive field.

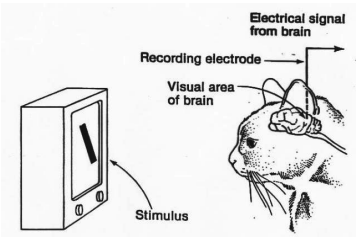


Fig 1. David Hubel and Torsten Wiesel probed the visual cortex of a cat with electrodes and found layers of cells that responded to local patterns of stimulation. (image widely used on the internet - source unknown)

Inspired by these results (and the subsequent Nobel Prize of Hubel and Weisel), computer vision researchers explored the use of image descriptions using convolution with Finite Impulse Response digital filters based on mathematical models of receptive fields [2], [3] including Gaussian derivatives and Gabor Functions [4]. Researchers in Machine learning speculated that it would be better to learn the weights for such filters with back-propagation. This would eventually lead to a new form of neural network known as a convolutional neural network. To properly understand such networks it can be worthwhile to review some basics from digital signal processing.

1.1 Convolution

Convolution describes the response of a linear time-invariant system to an input stimulus or driving function. An example of convolution is provided by shouting into a tunnel. The tunnel can be modeled as a shift invariant acoustic function that describe the multiple paths that sound waves of the shout may follow through the tunnel, with different lengths and different durations. The sound at the far end of the tunnel will be composed of multiple superimposed copies of the shout arriving at different times. An observer at the far end of the tunnel will hear a sonically blurred version of the shout. In addition, some vocal frequencies may resonate in the tunnel and dominant the

sound at the far end, while other frequencies may be attenuated by interference. The effect described mathematically as a convolution as shown in equation 1, where *s(t)* is the waveform of a sound, *f(t)* is a characteristic impulse response that describes the possible paths of the sound through the tunnel, and *u* is a dummy variable used for integration.

$$(s * f)(t) = \int_{-\infty}^{\infty} s(t-u) f(u)\, du \tag{1}$$

A copy of the waveform for the sound, *s(u)*, is placed at each time, *t*, and then multiplied by the characteristic impulse response of the tunnel, *f(u)*. For each time step, the resulting products are integrated and the result is placed in the output signal at position *t*. The result is a distorted version of the sound.

Computer science students generally find this operation easier to visualize and understand when expressed using sampled digitized signals. Let *s(n)* represent a sampled copy of the sound and *f(n)* represent the linear shift invariant system created by the tunnel. In order to compute the convolution it is necessary for at least one of the signals to have a finite duration. Let *N* represent the duration of the shorter of the two signal *s(n)* and *f(n)*. The discrete convolution equation is written as

$$(s * f)(n) = \sum_{m=0}^{N-1} s(n-m) f(m) \tag{2}$$

A copy of the sound *s(-)*, is placed at each time, *n*, and then scaled by the value of system, *f(n)*. For each value of *n*, the products are summed and the result is placed in the output signal. This is exactly the equation for convolution with a Finite Impulse Response (FIR) digital filter, *f(n)* composed of *N* coefficients with a digital signal, *s(n)*. Both multiplication and convolution are commutative, and so the order of the signals does not matter. In the engineering literature, convolution is commonly written as shown in equation 3.

$$f(n) * s(n) = \sum_{m=0}^{N-1} f(m) s(n-m) \tag{3}$$

Note that the operator "*" is exclusively reserved to represent convolution. This operator should never be used for multiplication in a context involving convolution.

For image processing, the image and filter are generally finite 2D signals with a positions defined over a range from 1 to N. For an image *P(x,y)* with the horizontal and vertical axes noted as *x* and *y*, the 2D convolution of an *NxN* filter *f(x,y)* would be written:

$$(f * P)(x,y) = \sum_{v=1}^{N} \sum_{u=1}^{N} f(u,v) P(x-u, y-v) \tag{4}$$

This operation can be seen as sliding the 2D filter, *f(x,y)* over the image and at each position, multiplying the weights of the filter *f(u,v)* by the pixels of the image, summing the product and placing this at the position *(x,y)*. Any image positions less than 1 or greater than the size of the image are taken as zero. The use of *x–u* and *y–v* rather

than $y+u$ and $x+v$ flips the filter around the zero position, resulting in a mirror image of the filter. This is a mathematical convenience to assure that convolution is equivalent to multiplication in the Fourier domain, and has no relevance to Convolutional Neural Networks. In the machine learning literature, it is not unusual to see authors neglect this detail and write $y+u$ and $x+v$.

In the 1980s, researchers in machine learning asked if such filters could not be learned using back-propagation[1]. It was observed that learning perceptrons for small local windows greatly reduces the number of parameters to learn, while greatly increasing the availability of training data. Training a single perceptron to provide a binary classification for 512 x 512 image would require learning 2^{18} parameters and each image would provide only one training sample for a binary decision. Alternatively, training a perceptron for an 8 by 8 receptive field would require learning only 257 parameters, and each 512 x 512 image could provide up to 2^{18} examples of local neighborhoods to use as training samples. This makes it practical to learn many layers of local perceptrons with several perceptrons at each level, much like the visual cortex of cats or humans. Such a network could be used to recognize many different classes of visual patterns.

The dominant paradigm in computer vision at the time (and until well after 2000) was that receptive fields should be designed as digital filters with well-defined mathematical properties for bandwidth and invariance to scale or rotation. However, one area where image processing with neural networks did show promise was in reading handwritten digits for mail sorting and check processing.

1.2 Early Convolutional Neural Networks: LeNet

In the early 1990s, the US National Institute of Standards and Technology (NIST) published a data set of digitized images of handwritten digits collected during the 1980 US census and issued a research challenge for recognizing handwritten digits using this data set. Such a technology could potentially be used to build machines for labor intensive tasks such as sorting mail and processing checks. A team of researchers at AT&T led by Yann Lecun began experimenting with neural networks architectures for this task. The team proposed a family of neural network architectures, referred to as LeNet, composed of multiple layers of receptive fields using a number of insights inspired by techniques used in image processing and computer vision [5]. A typical example of a LeNet architecture is shown in figure 2.

[1] Private discussion between the author and Geoff Hinton at CMU in 1982 or 1983.

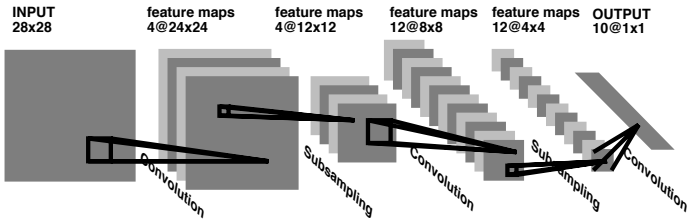
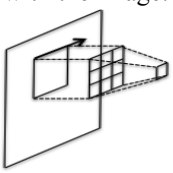


Fig 2. An early LeNet architecture for recognizing handwritten digits on checks and postal codes. Image copied from [5].

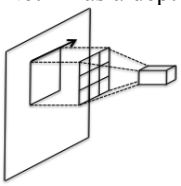
The first insight was to process the image as a set of 5x5 overlapping windows. Training a perceptron to process a 5x5 window requires learning only 26 parameters. Processing every position in the image with the same perceptron greatly increases the amount of data available for training, as each position of the image provides a training sample. Processing an image in this way is referred to as a "sliding window" detector. For a perceptron, the linear part of the perceptron is equivalent to convolving a digital filter with the image. This was referred to as a convolutional neural network.



$$a(i,j) = f\left(\sum_{u,v}^{N} w(u,v)p(i-u,j-v) + b\right) \qquad (4)$$

Fig 3. A convolutional network processes a stream of 2-D overlapping windows. In equation 4, *p(i,j)* is a 2D input layer, *w(u,v)* is an *NxN* learned receptive field, b is a learned bias, *f(-)* is a non-linear activation function as discussed in the chapter on training neural networks with back-propagation [6], and *a(i,j)* is the resulting output layer.

A second insight was to use several neural units in parallel to describe each window, as seen with the retinotopic maps observed in the visual cortex of the cat by Hubel and Weisel. This lead to a map of features for each pixel with the number of features referred to as the depth of the feature map. Figure 2 show that the first layer of LeNet-1 has a depth of 4.



$$a_d(i,j) = f\left(\sum_{u,v}^{N} w_d(u,v)p(i-u,j-v) + b_d\right) \qquad (5)$$

Fig 4. A convolutional network processes each window in parallel with D receptive fields, resulting in vector of D feature values $\vec{a}(i,j)$ for each image position *(i,j)*. Equation 5 generalizes equation 4 by replacing a single learned receptive field, *w(i,j)*, with a vector of D learned receptive fields, $w_d(i,j)$ generating a vector of d output layers, $a_d(i,j)$.

A third insight was to reduce the resolution of each layer by resampling and then processing the resulting resampled feature map with another convolutional network. For example the second layer of LeNet-1 was produced by subsampling the feature

map of first level using a sample distance of 2, and then processing the result with convolution by another set of 4 perceptrons trained with back-propagation. This resampling operation was referred to as "pooling" and had the effect of increasing the effective size of the receptive field at the second level, in a manner that is similar to the image pyramids used for computer vision at the time, and to the layers of larger receptive fields found in the deeper in the visual cortex of the cat by Hubel and Weisel. As the number of rows and columns of the feature map is reduced by successive resampling (or pooling), the number of features (depth) at each layer was increased. For example layers 3 and 4 of LeNet-1 contain features from convolution with 12 receptive fields (depth=12), as can be seen in figure 2. Once the image has been reduced to a 5x5 map of 16 features, the resulting 400 features are directly mapped by a perceptron to one of the 10 possible output class labels.

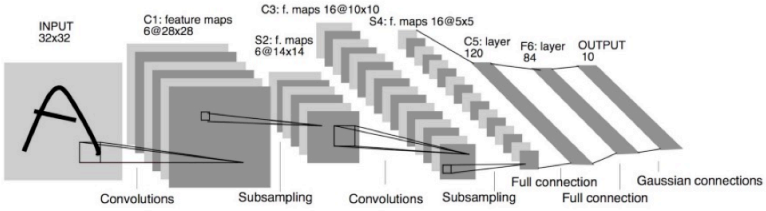


Fig 5. The LeNet-5 architecture presented at the 1997 IEEE CVPR [7].

The AT&T team experimented with several such architectures. The LeNet-5 architecture, shown in figure 5, was found to provide the best recognition rates for the NIST dataset of hand-written digits and was used to construct a commercially successful system for processing checks.

In order to describe the architectures for convolutional networks such as LeNet, we need to define some of the common "hyper-parameters" that are used to define convolutional networks.

1.3 Convolutional Network Hyper-parameters

Convolutional networks are commonly specified by a number of hyper-parameters. These include the Spatial Extent, Depth, Stride, Padding and Pooling:

Spatial Extent: This is the size of the filter. Early networks followed computer vision theory and used 11x11 or 9x9 filters. Experimentation has shown that 3x3 filters can work well with multi-layer networks.

Depth: This is the number *D* of receptive fields for each position in the feature map. For a color image, the first layer depth at layer 0 would be *D=3*. If described with 32 image descriptors, the depth would be *D=32* at layer 1. Some networks will use NxNxD receptive fields, including 1x1xD.

Stride: Stride is the step size, *S*, between window positions. By default, stride is generally set to 1, but for larger windows, it is possible define larger step sizes.

Zero-Padding: Size of region at the border of the feature map that is filled with zeros in order to preserve the image size (typically *N*).

Pooling: Pooling is a form of down-sampling that partitions the image into non-overlapping regions and computes a representative value for each region. An example of 2x2 max pooling is shown in figure 6. The feature map is partitioned into small non-overlapping rectangles, typically of size 2x2 or 4x4, and a single value it determined for each rectangle. The most common pooling operators are average and max. Median is also sometimes used. The earliest architectures used average pooling, where the neighborhood is replaced with the average value of the samples, creating a form of multi-resolution pyramid. Max pooling has generally been found to provide slightly better performance.

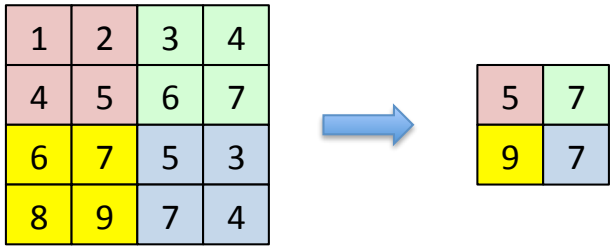


Fig 6. Max pooling replaces an NxN window of features with the largest feature value in the window. For example the 2x2 red square in the upper left corner is replaced by the largest of the 4 values in the square (5).

1.4 The LeNet-5 Architecture

LeNet-5 is composed of multiple repetitions of 3 operations: Convolution, Pooling, and Non-linearity. The system uses convolution of receptive fields of size 5x5 with a stride of 1, no zero padding and a depth of 6. Six receptive fields are learned for each pixel in the first layer. Using 5x5 filters without zero padding reduces the input window of 32x32 pixels to a layer of composed of 6 sets of 28x28 units. A sigmoid activation function was used for the activation function. Pooling was performed as a spatial averaging over 2x2 windows giving a second layer of 6x14x14. The output was then convolved with sixteen 5x5 receptive fields, yielding a layer with 16x10x10 units. Average pooling over 2x2 windows reduced this to a layer of 16x5x5 units. These were then fed to two fully connected layers and then smoothed with a Gaussian filter to produce 10 output units, one for each possible digit.

Despite the experimental and commercial success of LeNet, the approach was largely ignored by the computer vision community, which was more concerned at that time with multi-camera geometry and Bayesian approaches to recognition. The situation began to change in the early 2000's, driven by the availability of GPUs, and plan-

etary scale data, made possible by the continued exponential growth of the World Wide Web, and the emergence of challenge-based research in computer vision. During this period, computer vision and machine learning were increasingly organized around open competitions for performance evaluation for well-defined tasks using publically available "benchmark" data-sets.

Many of the insights of LeNet-5 continued to be relevant as more training data, and additional computing power enabled larger and deeper networks, as they allowed more effective performance for a given amount of training data and parameters.

2. Classic CNN Architectures

The emergence of the internet and the world-wide web made it possible to assemble large collections of training data with ground truth labels, and to issue global challenges for computer vision techniques for tasks such as image classification and object detection. Many of the most famous CNN architectures have been designed to compete in these large-scale image challenges, and the size of the input image and the number of output categories are often determined by the parameters of the challenge for which the network was designed.

Several key data sets that have influenced the evolution of the domain. Perhaps the most influential of these has been ImageNet.

ImageNet is an image database organized according to the nouns in the WordNet hierarchy compiled for research in Linguistics. In 2006, Fei-Fei Li began working on the idea for ImageNet based on the idea of providing image examples for each word in WordNet, eventually using Amazon Mechanical Turk to help with assigning WordNet words to images. The ImageNet data set was first presented as a poster at the 2009 Conference on Computer Vision and Pattern Recognition (CVPR) in Florida and later published in the Journal of Vision [8].

In 2010 Li joined with the European PASCAL Visual Object Class (POC) challenge team to create a joint research challenge on several visual recognition tasks. The resulting annual competition is known as the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). The ILSVRC uses a list of 1000 image categories or classes, including 90 of the 120 dog breeds classified by the full ImageNet schema. In 2010 and 2011, a good score for the ILSVRC top-5 classification error rate was 25%.

Winning teams during the first years used statistical recognition techniques such as Support Vector Machines (SVM) combined with image features such as Scale Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HoG). However, in 2012, Alex Krizhevsky won the competition with a deep convolutional neural net inspired by LeNet-5 called AlexNet, as shown in figure 7. AlexNet achieved an error rate of 16% (accuracy of 84%). This dramatic quantitative improvement marked the start of the rapid shift to techniques based on Deep Learning using Neural Networks by the computer vision community. By 2014, more than fifty institutions participated in the ILSVRC, almost exclusively with different forms of Network Architectures. In 2017, 29 of 38 competing teams demonstrated error rates less than 5% (better than 95% accuracy). Many state-of-the-art object detection networks now pre-train on

ImageNet and then rely on transfer learning to adapt the learned recognition system to a specific domain.

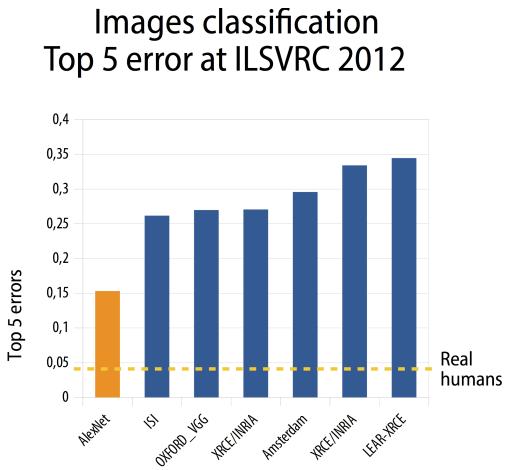


Fig 7. Error rates for the top 5 entries in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [9]

2.1 AlexNet

AlexNet [10], is a deeper and larger variation of LeNet5, using two parallel tracks of 5 convolutional layer followed by 3 fully connected layer. The initial receptive field is 11x11 with a stride (sample distance) of 4, followed by 48 parallel 5x5 receptive fields.

Innovations in AlexNet include:

1. **The use of ReLU avtivation instead of sigmoid or tanh**. ReLU provided a 6 times speed up with no loss of accuracy, allowing more training for the same cost in computation.
2. **DropOut**: A technique called “dropout” randomly chose units that are temporarily removed during learning. This was found to prevent over-fitting to training data.
3. **Overlap pooling**: Max pooling was performed with overlapping windows.

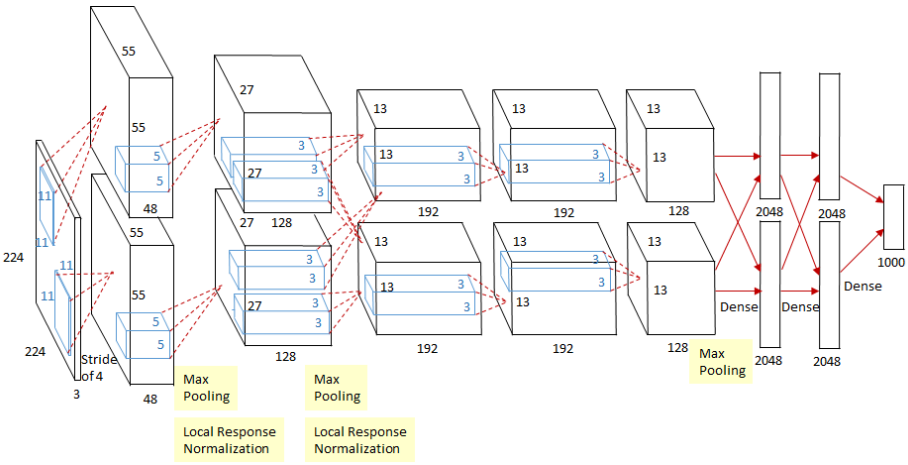


Fig. 7. The AlexNet architecture [10].
Image copied from https://medium.com/coinmonks/paper-review-of-alexnet-caffenet-winner-in-ilsvrc-2012-image-classification-b93598314160

The AlexNet architecture is composed of 5 convolutional layers followed by 3 fully connected layers. ReLU activation is used after each convolution and in each fully connected layer. The input image size of 224 x 224 is dictated by the number of layers in the architecture.

Source code for AlexNet can be found in PyTorch[2]. The network has 62.3 million parameters, and needs 1.1 billion computations in a forward pass. The convolution layers account for 6% of all the parameters, and consume 95% of the computation. The network is commonly trained in 90 epochs, with a learning rate 0.01, momentum 0.9 and weight decay 0.0005. The learning rate is divided by 10 once the accuracy reaches a plateau.

2.2 VGG-16 - Visual Geometry Group 16 Layer Architecture

In 2014, Karen Simonyan and Andrew Zisserman of the Visual Geometry Group at the Univ of Oxford demonstrated a series of networks referred to as VGG [11], shown in figure 8. An important innovation in VGG was the use of many small (3x3) convolutional receptive fields. VGG also introduced the idea of a 1x1 convolutional filter, using a perceptron to reduce the number of features (depth) at each image position. For a layer with a depth of D receptive fields, a 1x1 convolution performs a weighted sum of the D features, followed by non-linear activation using ReLU activation.

[2] An open source machine learning framework available at https://pytorch.org/.

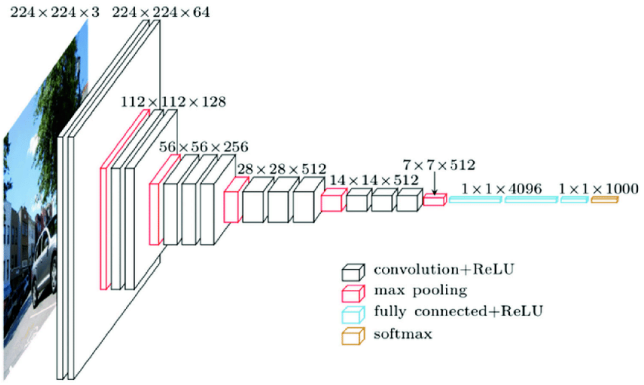


Fig. 8.The VGG-16 Architecture.

VGG uses a stack of 18 convolutional layers in which decreases in resolution provided by pooling are accompanied by increases in depth. The final 7x7x512 convolutional layer is followed by three Fully-Connected layers: the first two have 4096 channels while the third fully connected layer outputs a probability distribution for each of the 1000 classes of the ILSVR Challenge using a soft-max activation function. All except the last output layer use Relu activation.

2.3 YOLO: You Only Look Once

YOLO [12] poses object detection as a single regression problem that estimates bounding box coordinates and class probabilities at the same time directly from image pixels. This is known as a Single Shot Network (SSD). 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.

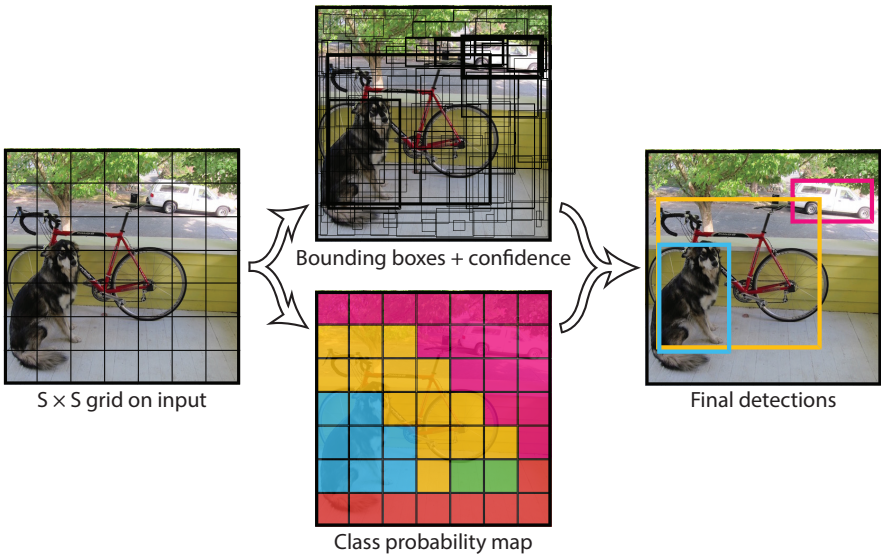


Fig 9. You Only Look Once (YOLO). The Yolo network simultaneously estimate bounding box coordinates and class probabilities for objects. Image copied from [12].

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). 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.

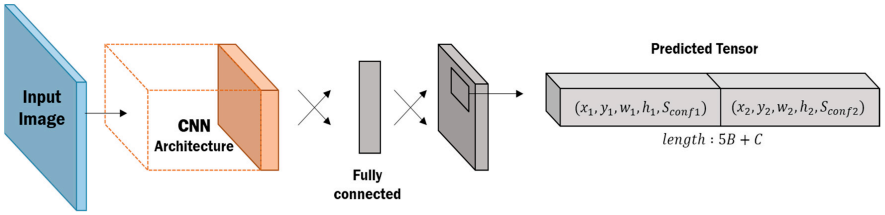


Fig. 10. Yolo uses a CNN architecture followed by a fully connected layer to simultaneously bounding boxes and classes for objects. Image copied from [13].

Each grid cell 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. The 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. Other-

wise the confidence score should equal the intersection over union (IOU) between the predicted box and the ground truth.

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.

The detection network has 24 convolutional layers followed by 2 fully connected layers as shown in figure 11. 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.

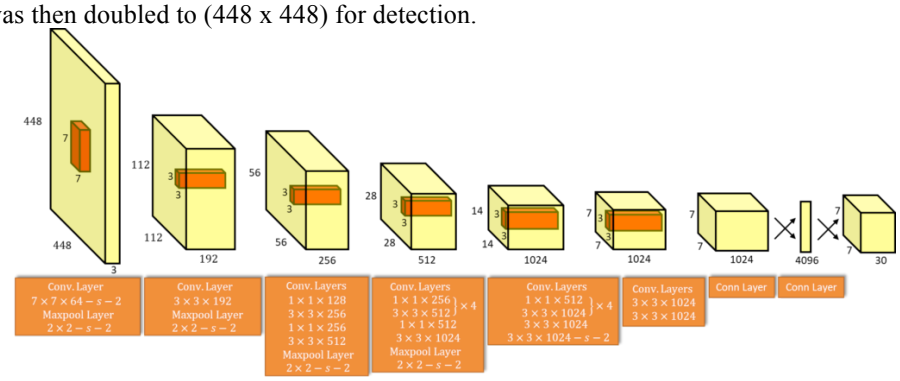


Fig 11. YOLO is composed of 24 convolutional layers followed by 2 fully connected layers. (from: http://datahacker.rs/how-to-peform-yolo-object-detection-using-keras/)

2.4 YOLO-9000 (YOLOv2)

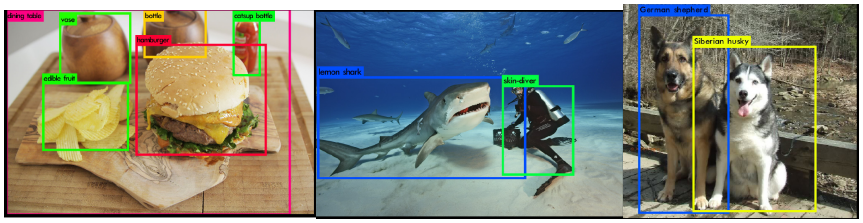


Fig 12. Typical output from YOLO-9000 [14].

In 2017, the YOLO team published performance evaluation results and source code for a new version of YOLO referred to as Yolo9000. Yolo9000 employed a number of innovations, including ideas that had emerged in the machine learning literature the previous year. At low resolutions YOLO9000 operates as a cheap, fairly accurate detector. At 288x288 it runs at more than 90 FPS. This makes it ideal for smaller GPUs, high frame rate 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

3. Conclusions

Convolutional neural networks are now a well established technology for analysis of multidimensional signals with applications in computer vision, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain–computer interfaces, financial time series and many other areas. New architectures for deep convolutional networks appear regularly addressing applications in an every expanding repertoire of domains.

Much of the progress of recent years has been obtained by training networks at ever-increasing depths, leveraging the growing availability of computing power provided by application specific integrated circuits and related technologies, and made possible by the availability of very large data set of annotated data. However, much of this work relied on supervised learning, and the need for annotated data has hindered development in some application domains.

Recently transformers, based on stacked layers of encoders and decoders with processing driven by self-attention, have begun to supplant convolutional neural networks in many areas, by improving performance while decreasing computational requirements. In addition, transformers can be trained by self-supervised learning, using data as its own ground truth, and eliminating the need for annotated training data as described in the chapter on Natural Language Processing with Transformers and Attention [15]. None-the-less, Convolutional networks remain an established tool with many practical applications.

Bibliography

1. Hubel, David. H., and Torsten N. Wiesel, Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, vol. 195, no 1, p. 215-243, 1968.
2. Rosenfeld, Azriel, *Picture Processing by Computer*, Academic Press, 1969.
3. Duda, Richard O. and Peter E. Hart, *Picture Processing and Scene Analysis*, Wiley, New York, 1973.
4. Zucker, Steven W., and Robert A. Hummel. *Receptive Fields and the Reconstruction of Visual Information*. New York Univ, Courant Institute of Mathematical Sciences, 1985.
5. LeCun Y., Jackel L., Bottou L., Brunot A., Cortes C., Denker J., Drucker H., Guyon I., Muller U.A., Sackinger E., Simard P., Comparison of learning algorithms for handwritten digit recognition. In *International Conference on Artificial Neural Networks*, Vol. 60, pp. 53-60, Nov 1995.
6. Crowley, James L., Machine Learning with Neural Networks, in *Advanced course on Human-Centered AI*, Editors Mohamed Chetouani, Virginia Dignum, Paul Lukowicz and Carles Sierra, Springer Lecture Notes in Artificial Intelligence (LNAI), 2022.
7. Bottou, Léon, Yoshua Bengio, and Yann LeCun. "Global training of document processing systems using graph transformer networks." In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,* CVPR 97. San Juan, Porto Rico, pp. 489-494, IEEE, 1997.
8. Fei-Fei, Li, Jia Deng, and Kai Li. "ImageNet: Constructing a large-scale image database." *Journal of vision* 9, no. 8: pp1037-1037, 2009

9. J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei. ILSVRC-2012.
10. Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).
11. Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." *arXiv preprint arXiv:1409.1556* (2014).
12. Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, CVPR 2016, IEEE, pp. 779-788. 2016.
13. Kim, Jinsoo, and Jeongho Cho. "Exploring a multimodal mixture-of-YOLOs framework for advanced real-time object detection." *Applied Sciences* 10, no. 2 p 612, 2020
14. Redmon, Joseph, and Ali Farhadi. "YOLO9000: better, faster, stronger." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263-7271. 2017.
15. Yvon, Francois, Natural Language Processing with Transformers and Attention, in *Advanced course on Human-Centered AI*, Editors Mohamed Chetouani, Virginia Dignum, Paul Lukowicz and Carles Sierra, Springer Lecture Notes in Artificial Intelligence (LNAI), 2022.