Chapter 4 – Image Modelling

Fabien Petit

Department of Economics, University of Barcelona
Centre for Education Policy and Equalising Opportunities, UCL

Introduction

The **feed-forward network** studied previously can also be used to model data in the form of (flattened) images

- The model has the ability to capture non-linearities and interactions among the input pixel intensities
- However, this is done by combining every intensity, which does not scale to larger (or deeper) images
- Pixel variables have both a spatial and spectral ordering, which can be used explicitly in the model

Introduction

Convolutional networks are a family of networks designed to process data in the form of multi-dimensional arrays

- They are called "convolutional" because some layers use a discrete convolution instead of a dot product
- This imposes more structure on the model, while reducing the number of parameters (i.e., special case)
- Applications include image regression and classification, as well as object localization and segmentation

Image Data

Image Data

A **digital image** is a representation containing an ordered set of numbers that a computer can process

- The image space is discretized into regular pixels containing quantized intensity values (e.g., $0 - 255$)
- Intensities represent brightness (i.e., grayscale), color (e.g., RGB), or other information (e.g., spectral)
- These values represent an average of the measured intensity over the pixel space (i.e., resolution)

Image Data

An image x_i is represented as a **multi-dimensional array** of pixel intensities with dimensions $k = h \times w \times d$

- The h and w dimensions record the coordinates of the pixels' position
- The d dimensions contain the pixel intensities (i.e., channels, bands)
- A single image observation has k "variables" (i.e., flattened image)

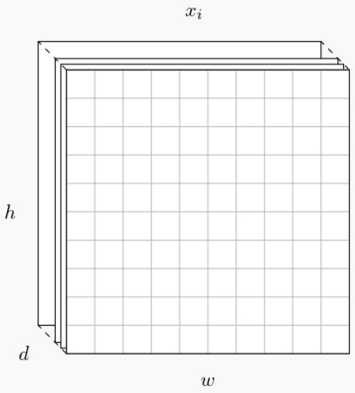


Image Data

Modelling images is challenging, the target function mapping pixel intensities to an output is complex and unknown

- **Dimensionality**: a color image with $k = 512 \times 512 \times 3$ has more than 786,000 intensities or variables
- **Interactions**: objects are represented by patterns of pixels (i.e., space) with specific intensities (i.e., channels)
- **Non-linearities**: there exists no simple mapping between the "color" information and the object

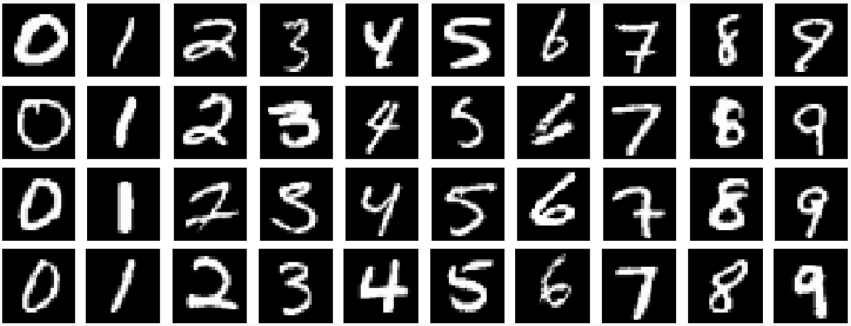
Image Data

To appreciate the complexity of image modelling, consider recognizing buildings from aerial images

- Objects in images vary in e.g., number, size, positions, appearance $\rightarrow$ intensities and spatial arrangement
- Most information is irrelevant (e.g., background), but minute details are important (e.g., other structures)

Image Data

Consider the optical character recognition task of recognizing the digits represented on grayscale images

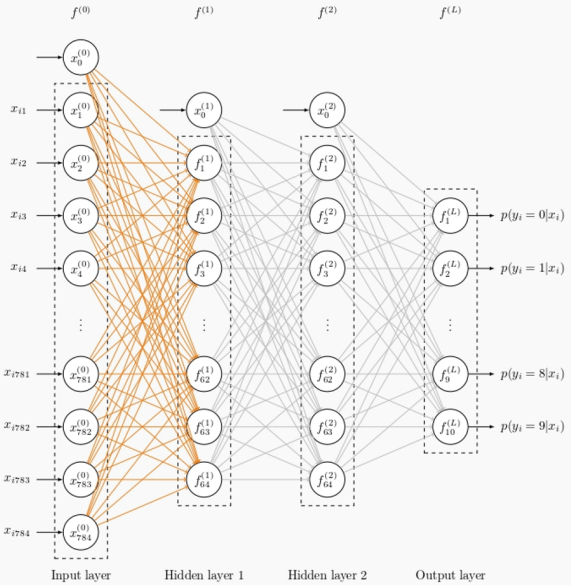


The MNIST handwritten digit database (LeCun et al. 2010) contains 60,000 images for training and 10,000 for testing. Each image $x_i^{(0)}$ has dimensions $k^{(0)} = 28 \times 28 \times 1$ and represents a single handwritten digits $y_i \in \{0, \ldots, 9\}$

Naive Approach

Naive Approach

Softmax and cross-entropy



One approach is to use the flattened vector of intensities $n \times \left(k^{(0)} \times h^{(0)} \times d^{(0)}\right)$ as input to a feed-forward network. However, interaction patterns are captured by combining every input intensity within each unit of the first hidden layer, which involves many parameters.

Details: Hidden and output layers use ReLU and softmax activation, respectively. The optimization minimizes the cross-entropy loss (i.e., negative log-likelihood of the multinomial distribution) and yields a test sample accuracy of 0.96.

Improvements

The **structure of the image** can be used explicitly to reduce the number of parameters while increasing accuracy

- **Local connectivity** across layers, each unit is connected to a small neighborhood in the previous layer
- **Shared parameters** across units, similar patterns are detected in different parts of the image
- **Multiple layers** efficiently capture increasingly complex interaction patterns at larger spatial scales

Convolution – Operation

Networks are convolutional when some of their layers use a discrete convolution in place of the dot product

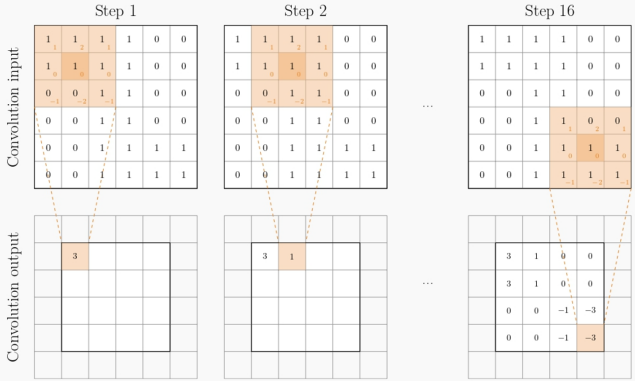
$$z_{id}^{(l)} = x_i^{(l-1)} * \beta_d^{(l)}$$

where $*$ is the convolution operation and $\beta_d^{(l)}$ is one of the $d^{(l)}$ convolutional kernels with dimensions $h_\beta \times w_\beta \times d^{(l-1)}$

- The discrete convolution captures interaction patterns between neighboring pixel intensities
- The parameters of the kernel correspond to a particular interaction pattern (e.g., edges, textures)

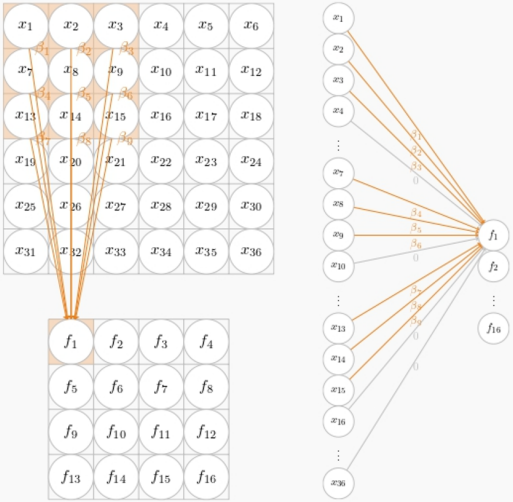
Convolution – Operation

Padding and stride



The kernel slides across input pixels, computing local sums of intensities weighted by its kernel parameters. The convolution operation is undefined on the edges. The kernel parameters correspond to one feature (i.e., interaction pattern), while its size defines the neighborhood over which this feature is computed.

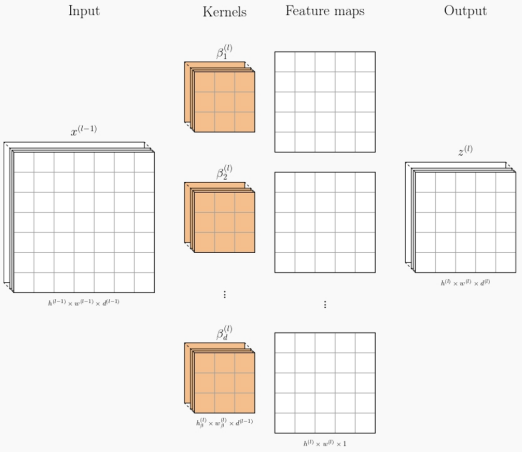
Convolution – Local Connectivity and Shared Parameters



- A convolution iteration is analogous to a local dot product in network units (i.e., z_u)
- A convolutional layer is a special case of a fully connected layer with shared parameters and many zero parameters (i.e., local connectivity)
- The kernel parameters are estimated using backpropagation, so the network learns which features to extract

Convolution – Convolutions with Multiple Channels

Depthwise separable convolutions



A convolutional layer applied to multi-channel images contains $d^{(l)}$ kernels of size $h_\beta^{(l)} \times w_\beta^{(l)} \times d^{(l-1)}$. Each kernel produces a feature map (i.e., grayscale image) whose intensity captures the presence or absence of a feature. Feature maps are stacked along the d dimension to produce the convolution output.

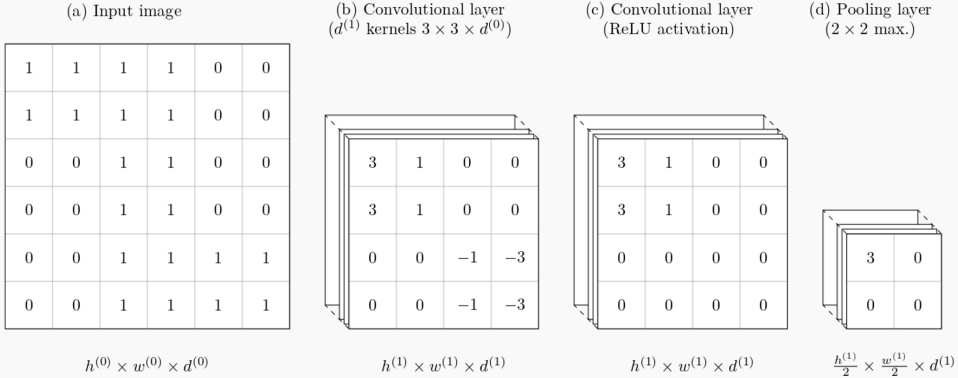
Convolution Networks

Pooling

Most convolutional networks use another operation called "**pooling**" to reduce dimensionality

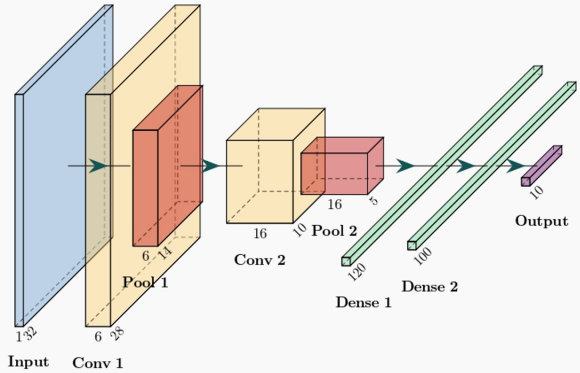
- Neighboring units often contain redundant information about the presence or the absence of a feature
- Pooling applies a summary function (e.g., mean, max) to a local patch of the convolution outputs (e.g., 2×2)
- Pooling also shifts high intensities shifted toward the center of the representation (i.e., robustness to location)

Convolution, Activation and Pooling



Convolution with unit stride and no padding. ReLU activation preserves edges at a particular orientation. A pooling layer reduces dimensionality by removing redundant patterns and improves location invariance. Pooling allows the next convolutional layer to detect interaction patterns on a larger spatial scale.

LeNet5 Convolutional Network (Lecun et al. 1998)



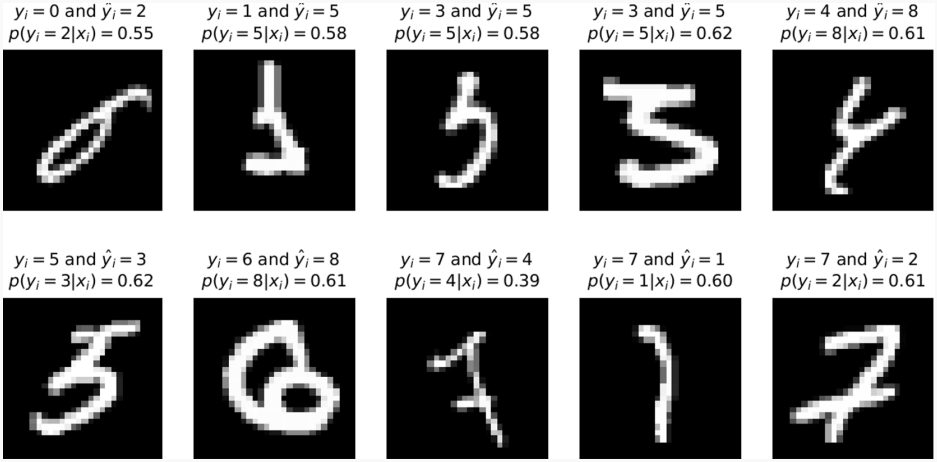
Consider a single image observation. Numerical structures are represented and connections across units are omitted. E.g. the first convolutional layer $f^{(1)}$ combines the input units in a $5 \times 5 \times d^{(0)}$ neighborhood using a set of $d^{(1)} = 6$ kernels. The pooling layer aggregates the intensities of each extracted representation in a 2×2 neighborhood.

Convolutional Networks – Structure

The **output of the last convolutional layer** can be fed to a feed-forward network (i.e., like in the naive approach)

- However, this layer has many fewer input units, each of which encodes meaningful features of the input image
- Convolutional layers perform feature extraction, while dense ones combine them into the predicted response
- Since the kernel parameters are estimated, the model decides which feature to extract (i.e., end-to-end)

Convolutional Networks – Sample of Wrongly Classified Observations



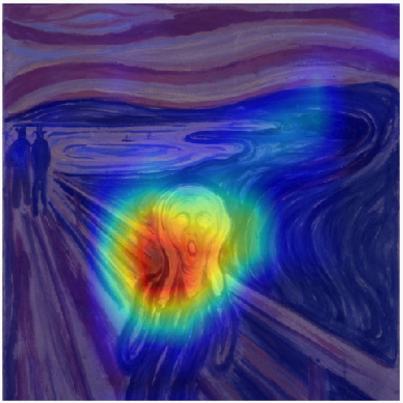
True and predicted responses and probabilities. The model represented in the LeNet5 model achieves 99.21% accuracy on the test sample. Experiment using the interactive tool (Harley 2015)

Interpretation

Saliency Map

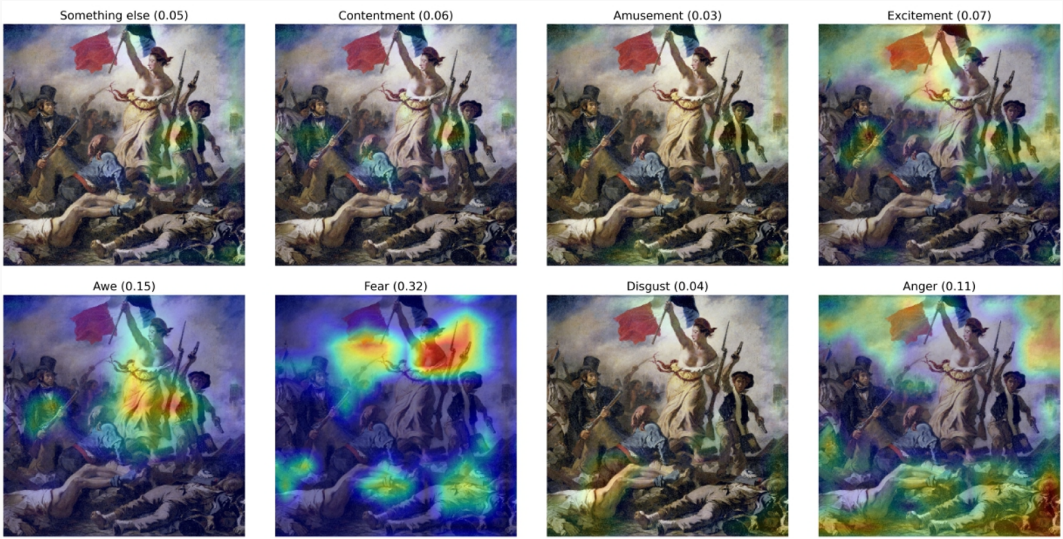
(a) Original Image

(b) Fear Saliency Map



Computed by summing the feature maps of a convolutional layer, weighted by their average gradient, over all spatial locations.

Emotion Saliency Map for Liberty Leading the People



Filter Visualization (Zeiler and Fergus 2014; Olah 2017)

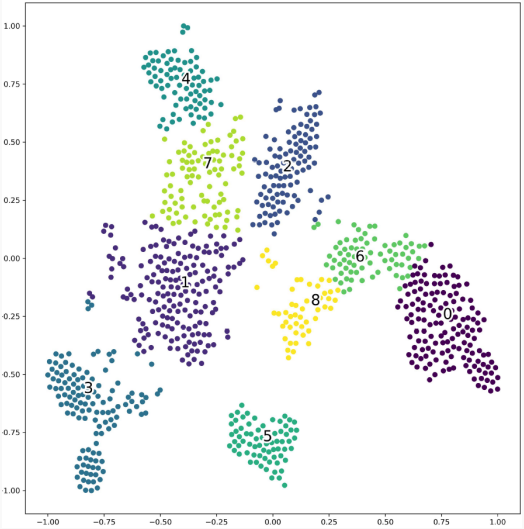
Computed by freezing the model parameters and computing the input image, maximizing the average score of a given feature map using gradient ascent.

- **Specialization**: different groups of units specialize in detecting distinct features (distributed representation)
- **Hierarchy**: Multiple convolutions capture features at increasingly large spatial scales (i.e., receptive field)

Feature Projections (Szegedy et al. 2015)

Each location of the image is characterized by scores capturing the presence or the absence of a feature.

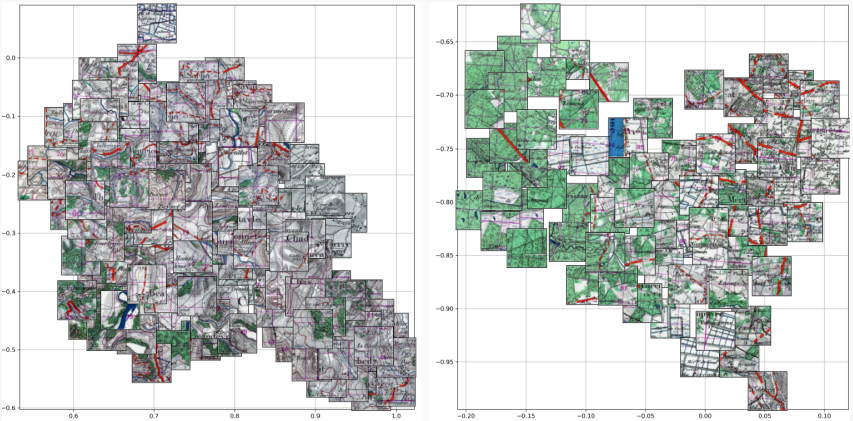
Representations – Latent Spaces



Two-dimensional T-SNE projection of average feature maps in the bottleneck block for test images (Mean shift clusters)

- Observations are mapped to numerical spaces where distances between pairs of points are well defined (Bengio et al. 2013)
- These representations disentangle the underlying factors of variation in the data (Wang et al. 2020)

Representations



Similar images have similar representations along multiple dimensions (e.g., legend, roads, land-use). Meaningful mathematical operations can be performed directly on these representation vectors, including distances (i.e,. inner product), averaging, and simple associations (addition, subtraction).

Representations

Using the mean representation of a test image (left), we subtract the mean representation of its cluster ($\approx$ map type) and add that of another cluster. The image closest to the resulting vector (right) represents a similar area (i.e., buildings, roads, rivers, grid) using another map type.

Optimization

Optimization – Dropout

At each optimization iteration, each unit is associated with a probability of being kept (Srivastava et al. 2014)

- Since the network can hardly rely on a particular input, it distributes more evenly the parameter values
- Dropout is not adapted to image data since features are captured using multiple neighboring units
- **Spatial dropout** (Tompson et al. 2015) randomly keeps entire representations along the $d^{(l)}$ dimension

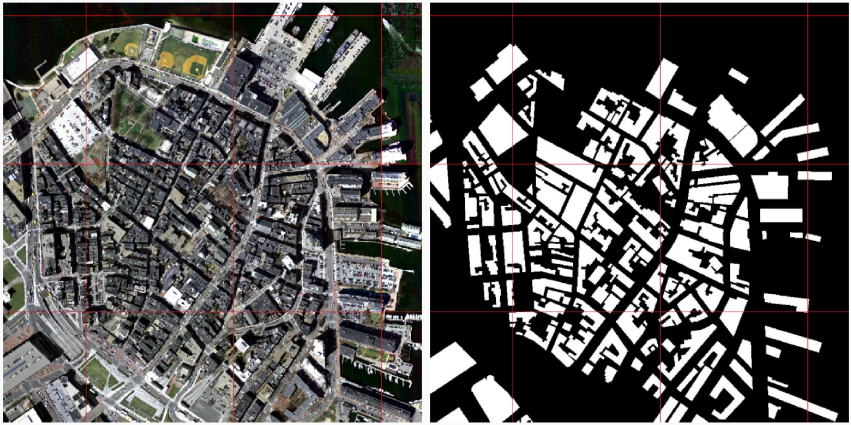
Augmentation

Augmentation artificially increases the training sample and enables the model to generalize better

- Random transformations are applied to each image in the batch before each optimization iteration
- Common operations involve flips, and parametrized shifts, rotations, zooms, and changes in brightness
- Parameters are drawn from a distribution of "reasonable" values, as the transformations must look credible

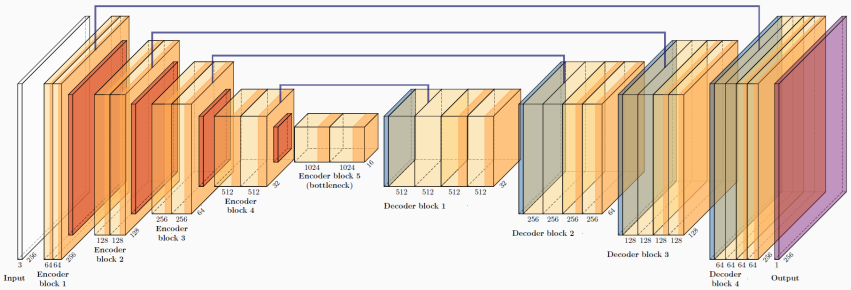
Segmentation

Segmentation of Buildings from Aerial Images



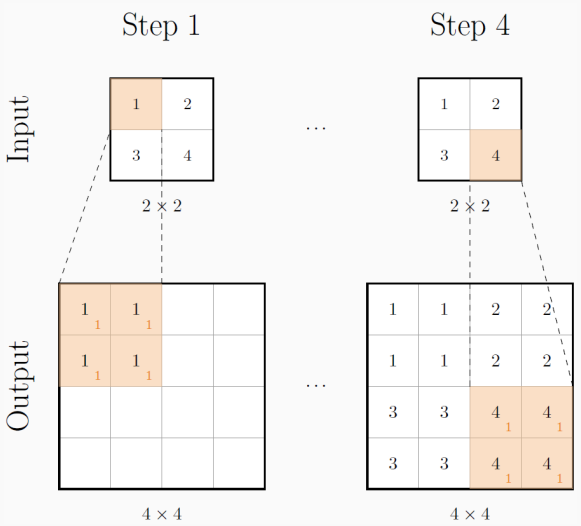
Semantic segmentation is a classification task at the pixel level. It aims at partitioning an image into multiple segments, each of which is a semantically meaningful region.

Segmentation – U-Net



U-Net architecture (Ronneberger et al. 2015). The 4 encoder blocks and the bottleneck contain two convolutional layers with $d^{(l)} \times 3 \times 3 \times d^{(l-1)}$ parameters and ReLU activation, followed by maximum pooling. The 4 decoder blocks combine the semantic (i.e., transposed convolution) and the fine-grained spatial information (i.e., skip connection) using two convolutions. The output layer uses a pointwise convolution with logistic activation. All convolutions have a unit stride and zero padding.

Transposed Convolution



Transposed convolution can be performed using a convolution in which the input has been padded with $w_\beta - 1$ and $h_\beta - 1$ pixels, respectively. This operation is less efficient than transposing the transformed kernel since it involves many zero multiplications

Segmentation – Skip Connections

A residual or **skip connection** is when a layer is connected to a layer precursor to its previous layer in the network

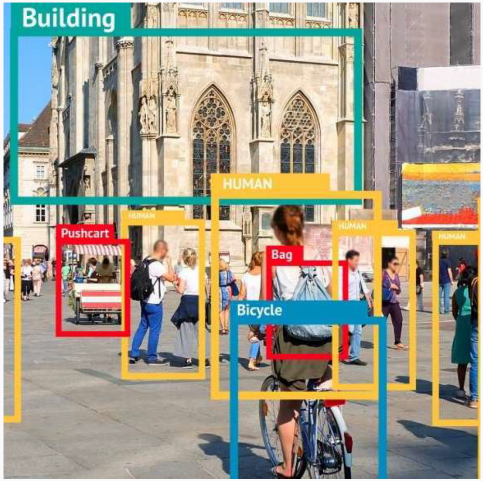
- The tensors are usually combined using concatenation (e.g., U-Net) or addition (e.g., ResNet)
- Improves optimization by mitigating the vanishing gradients problem, i.e., an alternative for backpropagation
- Adds an identity mapping to the output of a layer, which can help preserve important features

Localization

Localization

Object localization involves detecting, localizing, and classifying multiple objects in an image

- Multiple objects of different classes and shapes
- Objects can be located anywhere in the image
- Objects potentially overlapping or occluded
- Predict bounding box coordinates and class probabilities



YOLO

We study the **YOLO V1 (You Only Look Once) model** for object detection (Redmon et al. 2016)

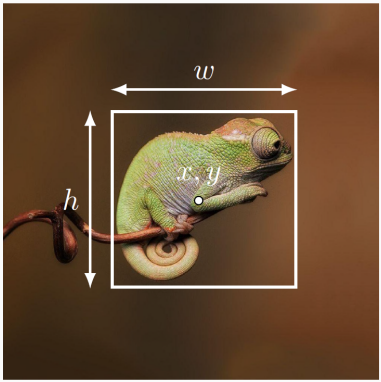
- Convolutional network integrates multiple object localization and classification in a single pass
- Extremely fast and relatively accurate predictions (e.g., real-time processing of video frames)
- Uses the entire image to compute robust representations and generalizes to other domains (e.g., paintings)

YOLO

Bounding boxes are represented numerically as the center coordinates, height, and width (x, y, h, w)

Quantities are expressed relatively to the image using the top-left corner as a reference point, i.e., $x, y, h, w \in [0, 1]$

$(0, 0)$



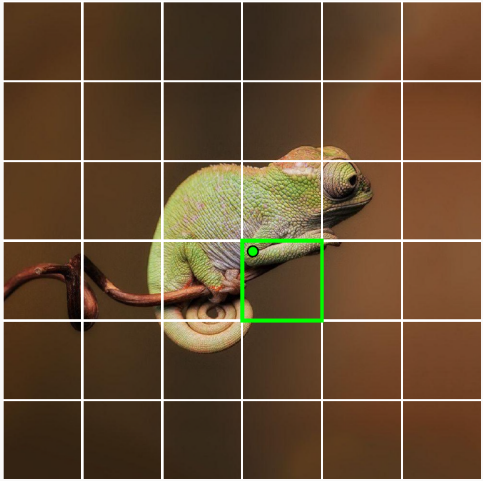
$(1, 1)$

YOLO

The image is partitioned into **cells**, each predicting a vector of object confidence, class probability, and box coordinates

$$y = \left[\underbrace{C}_{\text{object}}, \underbrace{p_1, p_2, \ldots, p_c}_{\text{class}}, \underbrace{x, y, w, h}_{\text{localization}} \right]$$

- When $C < T$ (threshold), the class and localization predictions are not meaningful
- For labels, objects spanning multiple cells are attributed to that contains their center point



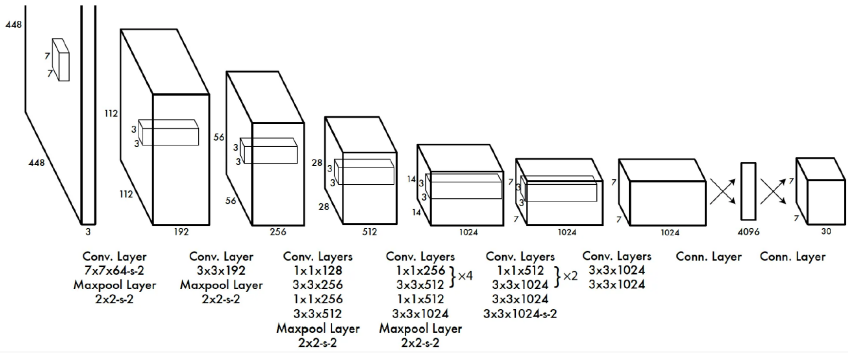
Anchor Boxes

A common issue is that a single cell can contain multiple objects,[1] potentially with different aspect ratios

- The model typically uses multiple anchor boxes with different aspect ratios, e.g., one tall, one wide
- In the labels, the object is attributed to the box with the highest overlap (i.e., IoU) with the ground truth
- This leads to specialization between the bounding box predictors (e.g,. sizes, aspect ratios, classes)

[1] YOLO V1 can predict a single object per cell, which is not the case in later versions.

YOLO V1 Architecture (Redmon et al. 2016)



See Szegedy et al. (2015) and Lin et al. (2013) for the design of the convolutional layers. The Pascal VOC dataset (Everingham et al. 2015) has 20 classes, and the model uses $S = 7$ and $B = 2$. The output tensor has dimensions $7 \times 7 \times (5 * 2 + 20)$. Each cell can only detect a single object. Layers use the leaky ReLU activation function.

Loss

The **YOLO loss function** balances localization, confidence, and class losses (but sum-squared errors are not always sensible)

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

Indicators $\mathbb{1}_i^{obj}$ denote if an object appears in cell i and $\mathbb{1}_{ij}^{obj}$ whether the bounding box j in cell i is responsible. Tuning parameters $\lambda_{\text{coord}} = 5$ increases the importance of bounding box predictions and $\lambda_{\text{noob } j} = .5$ decreases that of confidence predictions for empty boxes. The square root is used so that small deviations in large boxes matter less than in small boxes.

Non-Max. Suppression

Another issue is that multiple bounding boxes usually predict the same object, with varying degrees of confidence

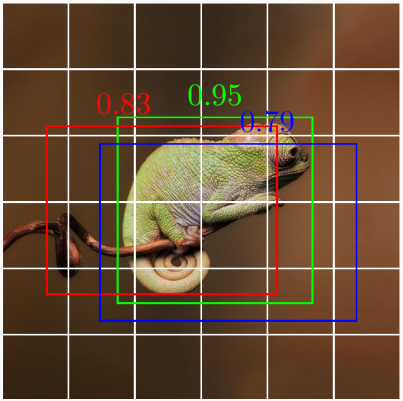
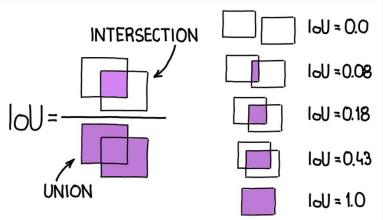


Figure: Intersection over Union



The IoU metric measures the quality of the overlap between the predicted bounding box and the ground truth

Non-Max. Suppression

For each class and object in the image, non-maximal suppression selects the best bounding boxes

Algorithm 1 Non-Maximum Suppression Algorithm

Require: Set of predicted bounding boxes B, confidence scores S, IoU threshold τ, confidence threshold T
Ensure: Set of filtered bounding boxes F

$F \leftarrow \emptyset$
Filter the boxes: $B \leftarrow \{b \in B \mid S(b) \geq T\}$
Sort the boxes B by their confidence scores in descending order
while $B \neq \emptyset$ **do**
    Select the box b with the highest confidence score
    Add b to the set of final boxes F: $F \leftarrow F \cup \{b\}$
    Remove b from the set of boxes B: $B \leftarrow B - \{b\}$
    for all remaining boxes r in B **do**
        Calculate the IoU between b and r: $iou \leftarrow IoU(b, r)$
        if $iou \geq \tau$ **then**
            Remove r from the set of boxes B: $B \leftarrow B - \{r\}$
        end if
    end for
end while

1. Select proposal b with $\max(S)$
2. Remove proposals r when $\text{IoU}(b, r) > \tau$
3. Iterate until all boxes have been processed

Summary

Summary

Convolutional networks are a family of networks designed to process (flattened) image data

- Applications include image regression, classification, localization, and segmentation
- A digital image is represented as a multi-dimensional array of pixel intensities with large dimensions
- Layers use a discrete convolution, instead of the dot product, which captures interaction patterns

References

References

- Lecun, Y. et al. (1998). "Gradient-based learning applied to document recognition". In: Proceedings of the IEEE 86.11, pp. 2278-2324 (cit. on p. 22).
- LeCun, Yann, Corinna Cortes, and Christopher C.J. Burges (2010). "MNIST handwritten digit database". In: ATT Labs (cit. on p. 10).
- Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). "Representation learning: A review and new perspectives". In: IEEE Transactions on Pattern Analysis and Machine Intelligence 35.8, pp. 1798-1828 (cit. on p. 33).
- Lin, Min, Qiang Chen, and Shuicheng Yan (2013). "Network In Network". In: CoRR abs/1312.4400 (cit. on p. 50).
- Srivastava, Nitish et al. (2014). "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: Journal of Machine Learning Research 15.56, pp. 1929-1958 (cit. on p. 37).
- Zeiler, Matthew D. and Rob Fergus (2014). "Visualizing and understanding convolutional networks". In: Computer Vision - ECCV 2014, pp. 818-833 (cit. on p. 30).
- Everingham, M. et al. (2015). "The Pascal visual object classes challenge: A retrospective". In: International Journal of Computer Vision 111.1, pp. 98-136 (cit. on p. 50).

References

- Harley, Adam W. (2015). "An interactive node-link visualization of convolutional neural networks". In: Advances in Visual Computing. Ed. by George Bebis et al. Springer International Publishing, pp. 867-877 (cit. on p. 26).
- Nielsen, Michael A. (2015). Neural networks and deep learning. Determination Press.
- Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015, pp. 234-241 (cit. on p. 41).
- Szegedy, C. et al. (2015). "Going deeper with convolutions". In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1-9 (cit. on pp. 31, 50).
- Tompson, Jonathan et al. (2015). "Efficient object localization using Convolutional Networks". In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 648-656 (cit. on p. 37).
- Dumoulin, Vincent and Francesco Visin (2016). "A guide to convolution arithmetic for deep learning". In: ArXiv e-prints (cit. on p. 67).
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). Deep Learning. MIT Press.
- Redmon, J. et al. (2016). "You only look once: Unified, real-time object detection". In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788 (cit. on pp. 46, 50).

References

- Chollet, François (2017). "Xception: Deep learning with depthwise separable convolutions". In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1800-1807 (cit. on p. 70).
- Olah, Christopher (2017). "Feature visualization". In: Distill (cit. on p. 30).
- Wu, Jianxin (2017). "Introduction to convolutional neural networks". Lecture notes.
- Wang, Jiayun et al. (2020). "Orthogonal convolutional neural networks". In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (cit. on p. 33).
- Zhang, Aston et al. (2023). Dive into Deep Learning. https://D2L.ai. Cambridge University Press.

Appendix

Go back

The multinomial logistic or softmax activation function measures $P\left(y_i = class \mid z_i\right)$ and can be written as

$$\sigma_{sm}\left(z_{iu}\right) = \frac{e^{z_{iu}}}{\sum_{u=1}^{k^{(L)}} e^{z_{iu}}} \tag{1}$$

where z_{iu} represents the result of the dot product in each output unit. The cross-entropy loss function is

$$\mathcal{L}_i\left(y_i, \hat{y}_i\right) = -\sum_{u=1}^{k^{(L)}} y_i \log\left(\hat{y}_{iu}\right) \tag{2}$$

where $k^{(L)}$ is the number of possible responses and $\hat{y}_{iu} = \sigma_{sm}^{(L)}\left(z_{iu}^{(L)}\right)$ is the predicted class probability

Padding and Stride

Padding involves adding extra pixels around the input image's edges to maintain the spatial dimensions of the convolution

- Add $\left\lfloor \frac{h_\beta - 1}{2} \right\rfloor$ rows above and $\left\lfloor \frac{h_\beta}{2} \right\rfloor$ rows below
- Add $\left\lfloor \frac{w_\beta - 1}{2} \right\rfloor$ cols. on the left and $\left\lfloor \frac{w_\beta}{2} \right\rfloor$ cols. on the right

where $h \times w$ and $h_\beta \times w_\beta$ are the image and kernel dimensions

- Padding helps preserve important information at the borders, pixels are usually set to 0
- Helps model design by facilitating the computation of the dimensions (e.g. residual connections)

Padding and Stride

Go back

Stride refers to the number of pixels by which the convolutional filter moves across the input image

Stride larger than one can be used as a (parametrized) dimensionality reduction operation. The size of the convolution output can be computed using

$$h^{(l)} = \left(\frac{h^{(l-1)} - h_{\beta}^{(l)} + 2p}{s} + 1 \right) \quad w^{(l)} = \left(\frac{w^{(l-1)} - w_{\beta}^{(l)} + 2p}{s} + 1 \right)$$

where s denotes the stride and p the padding. See Dumoulin and Visin (2016) for convolution arithmetics

Convolution as a Matrix Product

Consider the following convolution operation

$$\begin{bmatrix} 1 & 2 & 3 & 1 \\ 4 & 5 & 6 & 1 \\ 7 & 8 & 9 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 12 & 16 & 11 \\ 24 & 28 & 17 \end{bmatrix}$$

Convolution as a Matrix Product

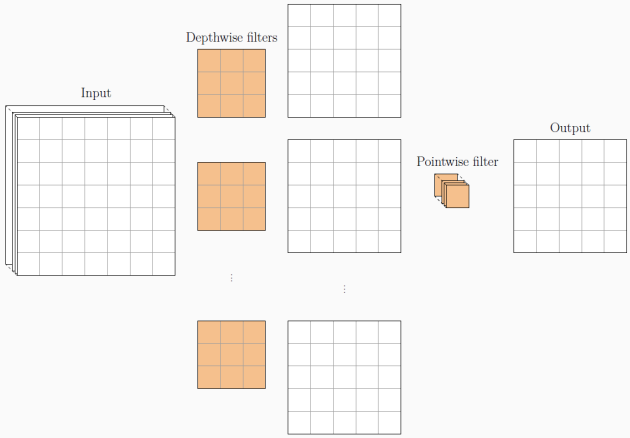
Go back

$$\text{reshape}(X) = \begin{bmatrix} 1 & 4 & 2 & 5 & 3 & 6 \\ 4 & 7 & 5 & 8 & 6 & 9 \\ 2 & 5 & 3 & 6 & 1 & 1 \\ 5 & 8 & 6 & 9 & 1 & 1 \end{bmatrix} \quad \text{flatten}(\beta) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

$$X' \cdot \beta = \begin{bmatrix} 12 \\ 24 \\ 16 \\ 28 \\ 11 \\ 17 \end{bmatrix} \quad \text{reshape}\left(X' \cdot \beta\right) = \begin{bmatrix} 12 & 16 & 11 \\ 24 & 28 & 17 \end{bmatrix}$$

Depthwise Separable Convolutions

Go back



To reduce the number of parameters, Chollet (2017) performs depthwise separable convolutions, which uses $d^{(L-1)}$ convolutions with kernels of size $h_\beta \times w_\beta \times 1$ combining the spatial information, and a pointwise convolution using a $1 \times 1 \times d^{(L-1)}$ kernel combining the features.

Generalization

If the input has dimension $H^{(l)} \times W^{(l)} \times D^{(l)}$ and kernel has dimension $h_\beta \times w_\beta \times D^{(k)}$, then the output has dimension $\left(H^{(l)} - h_\beta + 1\right) \times \left(W^{(l)} - w_\beta + 1\right) \times D$

- We add $\left\lfloor \frac{h_\beta - 1}{2} \right\rfloor$ rows above and $\left\lfloor \frac{h_\beta}{2} \right\rfloor$ rows below
- We add $\left\lfloor \frac{w_\beta - 1}{2} \right\rfloor$ columns on the left and $\left\lfloor \frac{w_\beta}{2} \right\rfloor$ columns on the right

Padded pixels are usually set to zero, but other values are possible