Image analysis

4. Convolutional networks

Clément Gorin
clement.gorin@univ-paris1.fr

Sorbonne School of Economics
Masters in Development Economics

Introduction

Introduction

The dense network studied previously can be applied to model data in the form of flattened images

- This model has the ability to capture non-linearities and interactions among the pixel variables
- However each hidden unit combine every pixel value, which does not scale to larger or deeper images
- Image pixels have a spatial and a spectral ordering, which should be exploited explicitly in the model

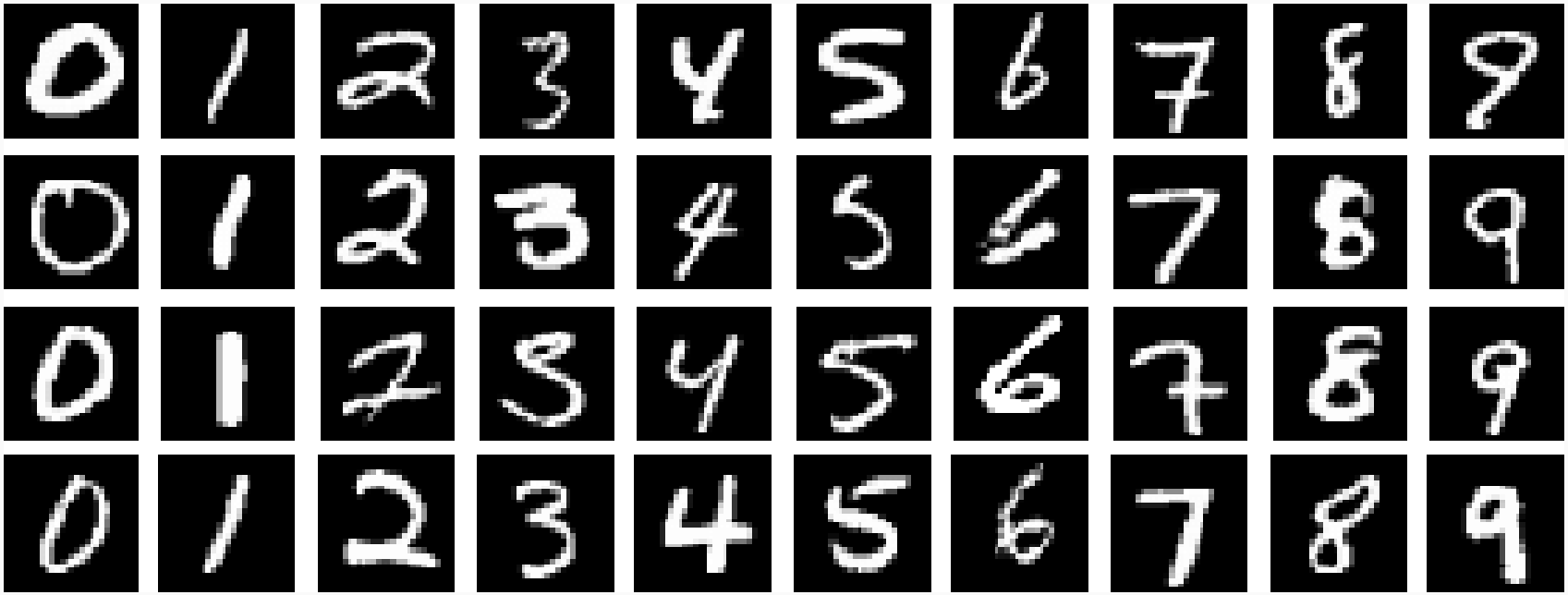
Introduction

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 additional structure on the model, while reducing the number of parameters (i.e. shared)
- Common applications include image regression and classification, object localisation and segmentation

Introduction

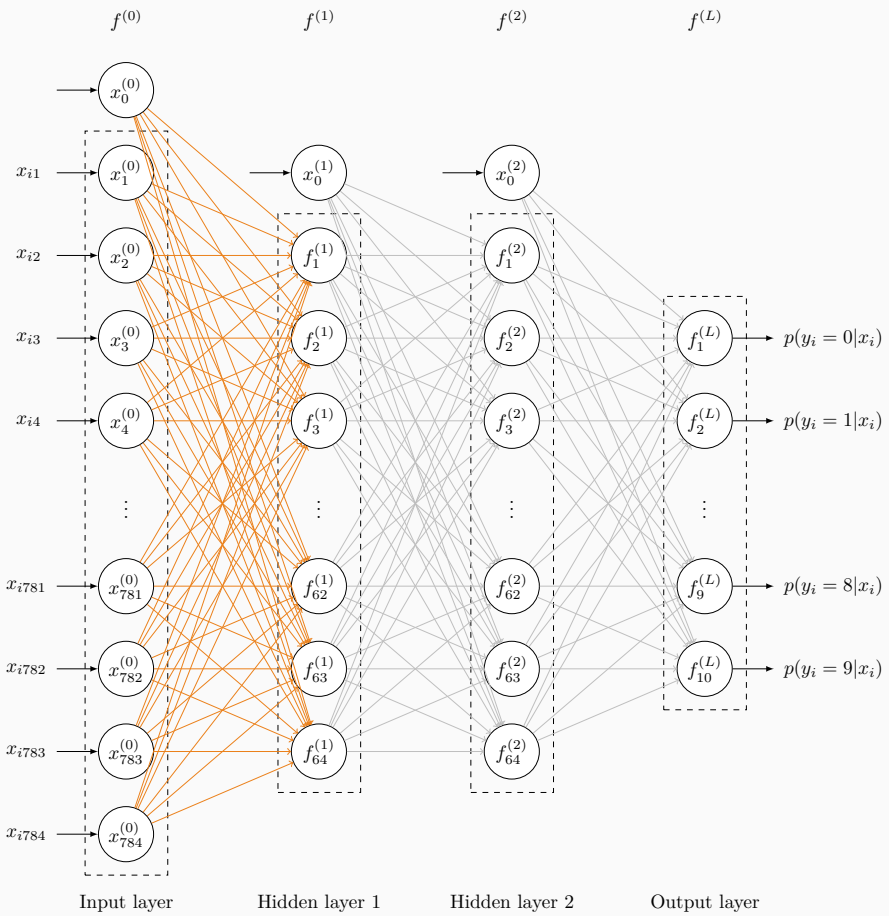
Consider the optical character recognition task of recognising the digits represented on greyscale images



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



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 optimisation minimises the cross-entropy loss (i.e. negative log-likelihood of the multinomial distribution) and yields a test sample accuracy of 0.96.

▸ Softmax and cross-entropy

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 small neighbourhood in the previous layer
- **Shared parameters** across units, similar patterns are detected in different parts of the image
- **Multiple layers** efficiently capture increasingly complex interactions patterns at larger spatial scales

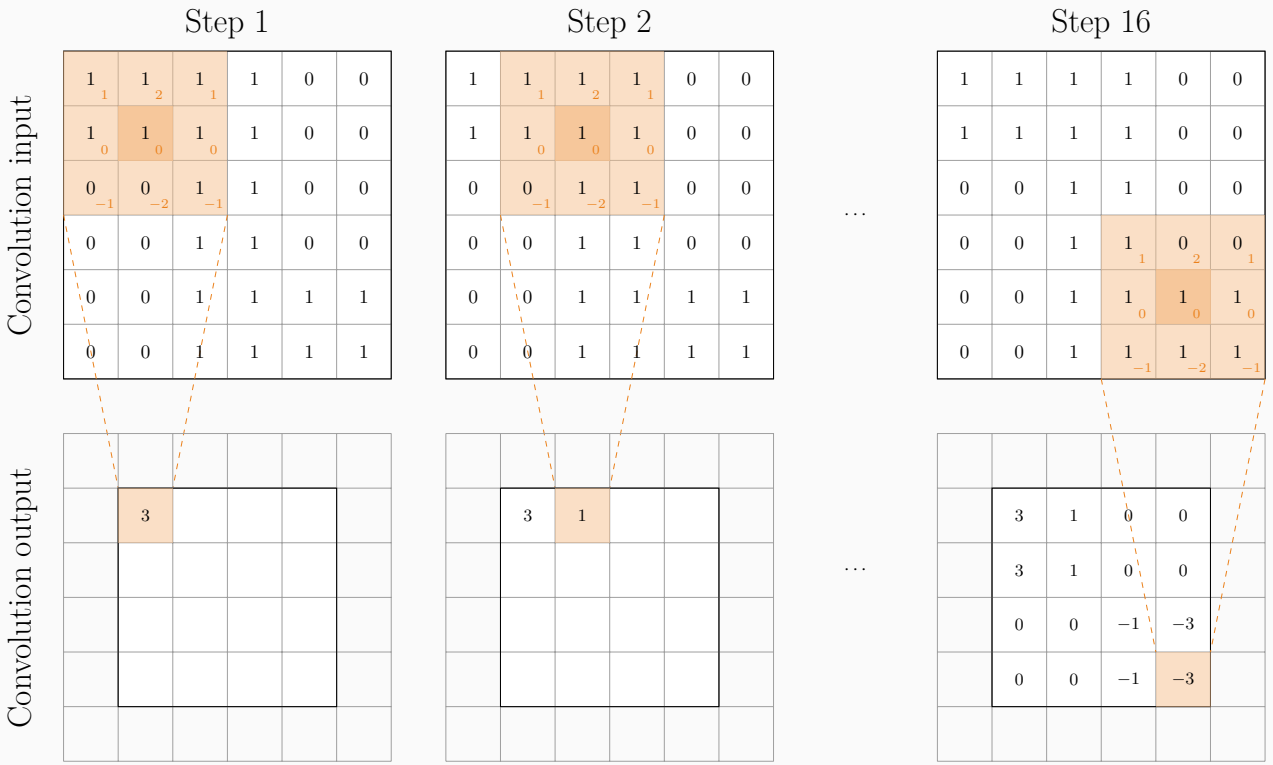
Convolution

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 $h_\beta \times w_\beta \times d^{(l-1)}$

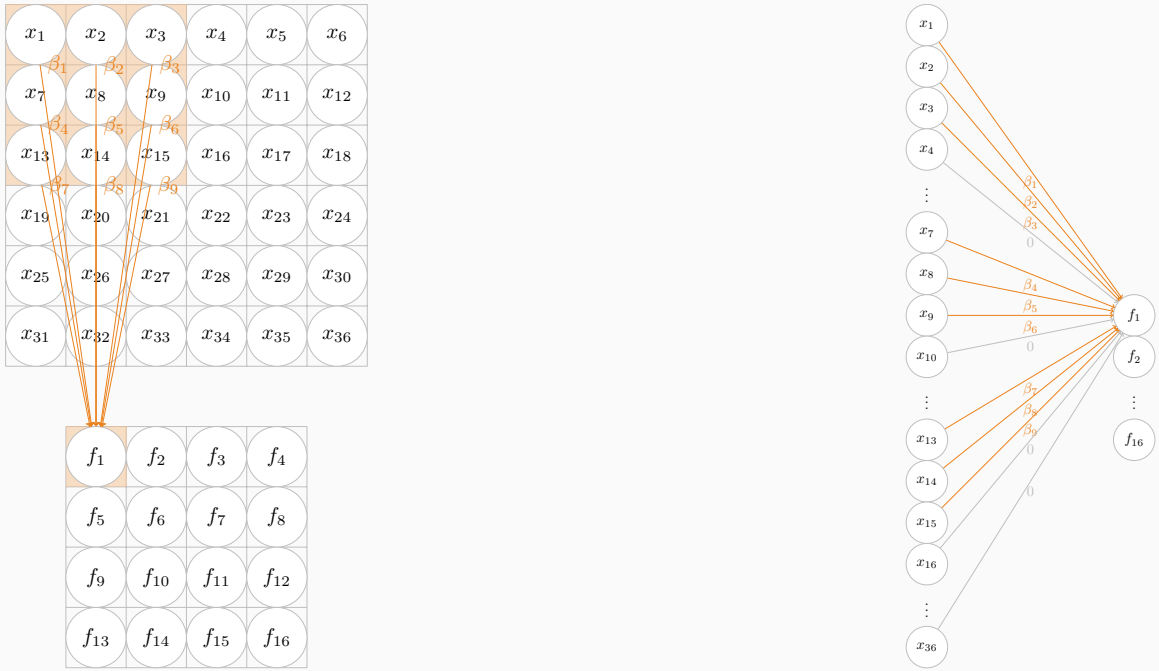
- The discrete convolution captures interaction patterns between neighbouring pixel intensities
- The parameters of the kernel correspond to a particular interaction pattern (e.g. edges, textures)



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 neighbourhood over which this feature is computed.

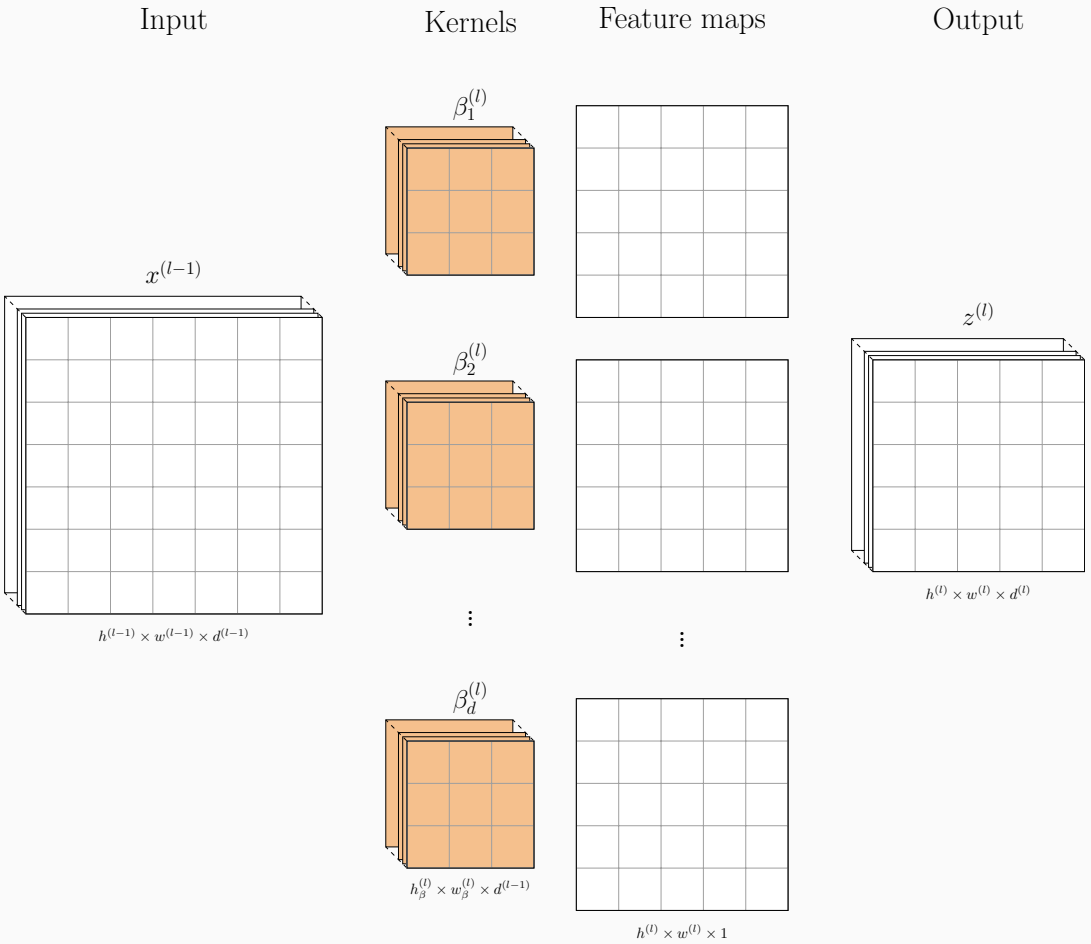
▸ Padding and stride

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 as a matrix product



Convolutions with multiple channels

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. greyscale image) whose intensity capture the presence of absence of a feature. Feature maps are stacked along the d dimension to produce the convolution output.

▸ Depthwise separable convolutions

Convolutional networks

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

- Neighbouring 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 centre of the representation (i.e. robustness to location)

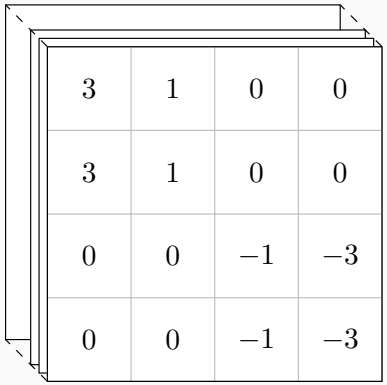
Convolution, activation and pooling

(a) Input image

1	1	1	1	0	0
1	1	1	1	0	0
0	0	1	1	0	0
0	0	1	1	0	0
0	0	1	1	1	1
0	0	1	1	1	1

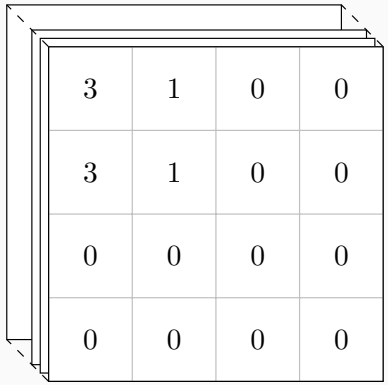
$h^{(0)} \times w^{(0)} \times d^{(0)}$

(b) Convolutional layer ($d^{(1)}$ kernels $3 \times 3 \times d^{(0)}$)



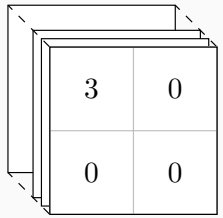
$h^{(1)} \times w^{(1)} \times d^{(1)}$

(c) Convolutional layer (ReLU activation)



$h^{(1)} \times w^{(1)} \times d^{(1)}$

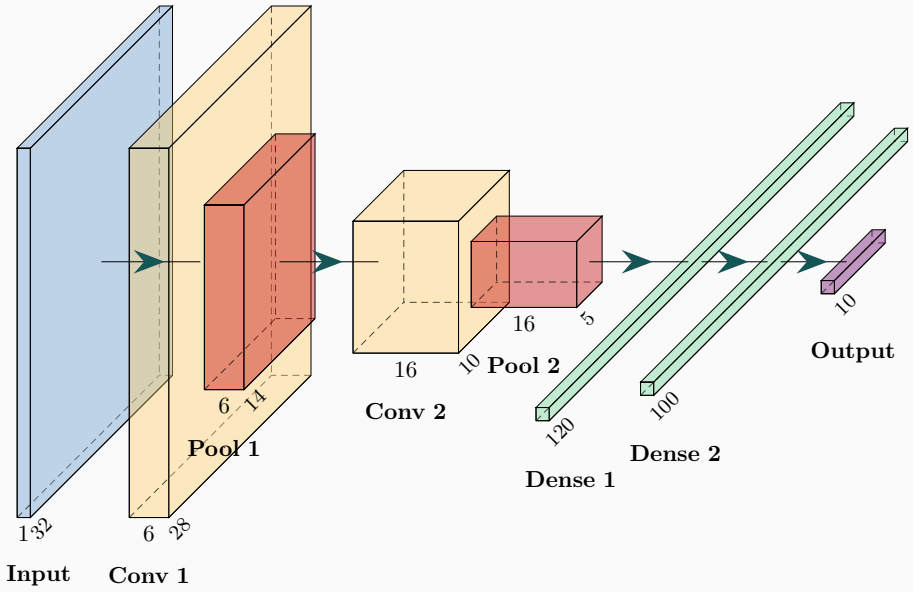
(d) Pooling layer (2×2 max.)



$\frac{h^{(1)}}{2} \times \frac{w^{(1)}}{2} \times d^{(1)}$

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

LeNet5 convolutional network (Lecun et al. 1998)

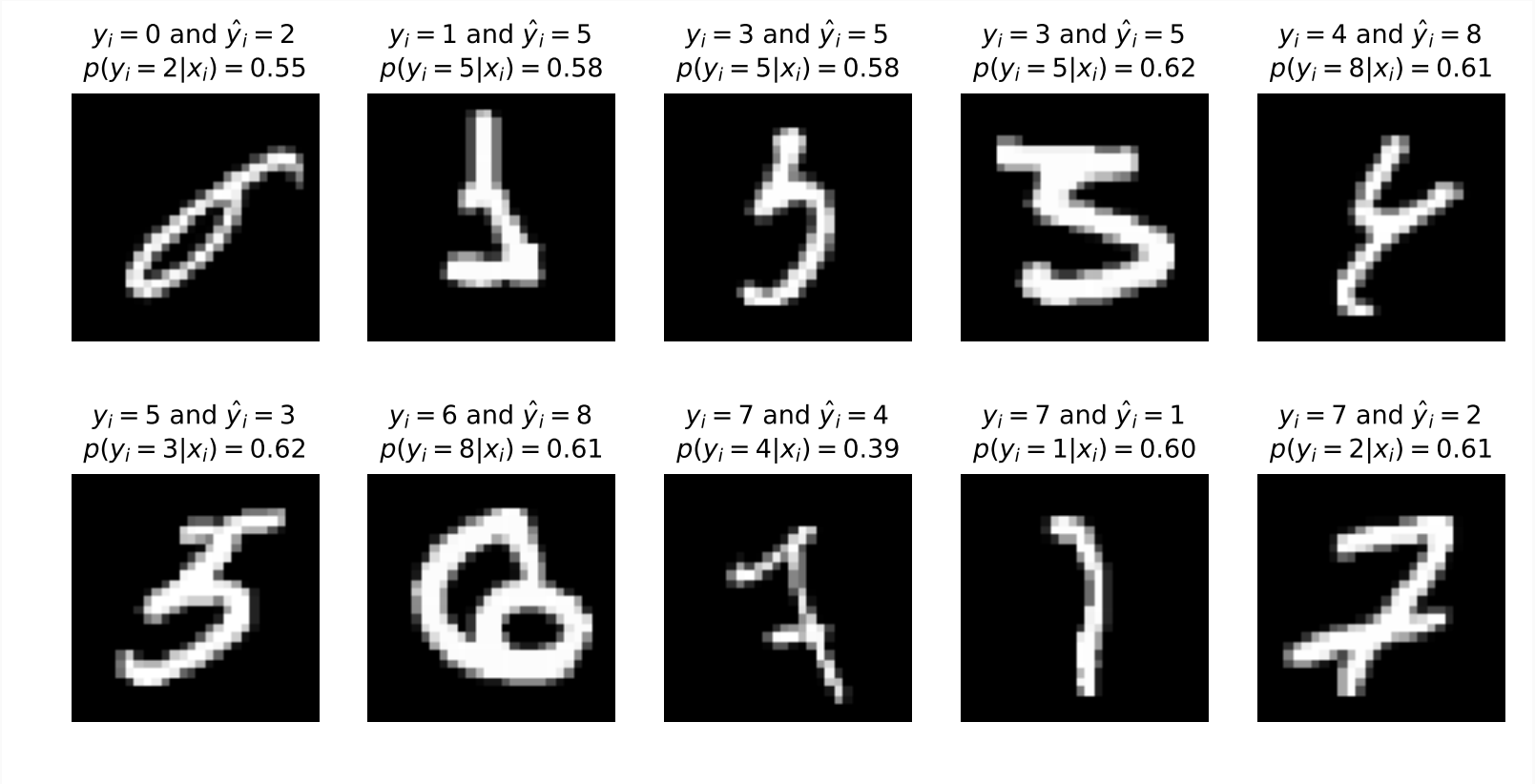


Consider a single image observation. Numerical structures are represented and connections across units omitted. E.g. the first convolutional layer $f^{(1)}$ combines the input units in a $5 \times 5 \times d^{(0)}$ neighbourhood using a set of $d^{(1)} = 6$ kernels. The pooling layer aggregates the intensities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

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

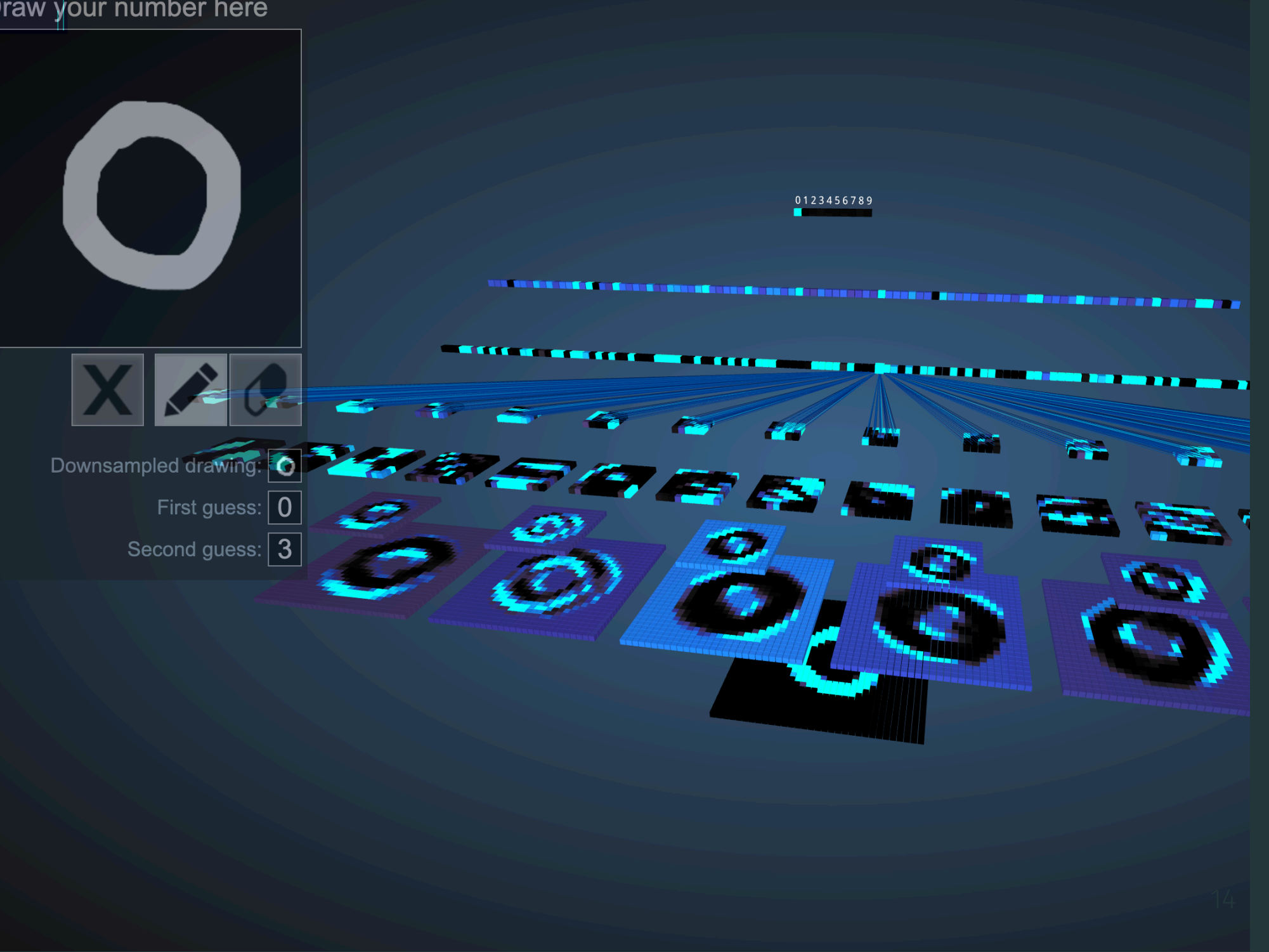
- However, this layer has much fewer input units, each of which encodes meaningful features of the input image
- Convolutional layers perform feature extraction, while dense ones combine them into the predicted response
- Since the kernel parameters are estimated, the model decides which feature to extract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

Sample of wrongly classified observations



True and predicted response and probabilities. The model represented on the LeNet5 model achieves 99.21% accuracy on the test sample.

Interpretation


raw your number here
0123456789
Downsampled drawing:
First guess: 0
Second guess: 3

Grad-CAM (Selvaraju et al. 2017) is used to visualise which regions the input influences a model's prediction

$$\text{CAM}_c = \text{ReLU}\left(\sum_k \alpha_k^c A^d\right)$$

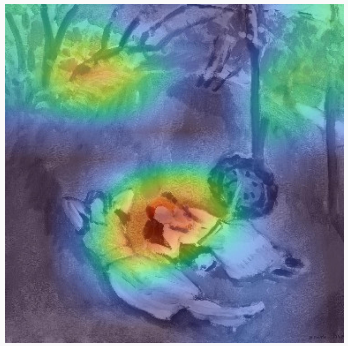
$$\alpha_d^c = \frac{1}{hw}\sum_i\sum_j \frac{\partial y^c}{\partial A_{ij}^d}$$

where A^d is the d-th feature map of the last convolutional layer, and α_d^c represents its importance for class c

Grad-CAM saliency map for fear

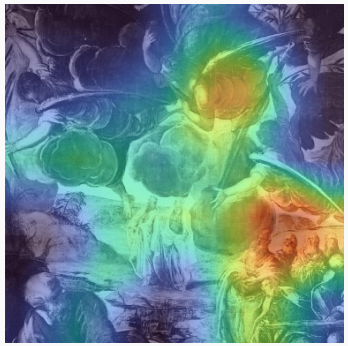
Weighted sum of the feature maps of the last convolutional layer, where the weights are the average gradients of the target class score with respect to each feature map.

Saliency maps for dominant emotions


Contentment


Amusement

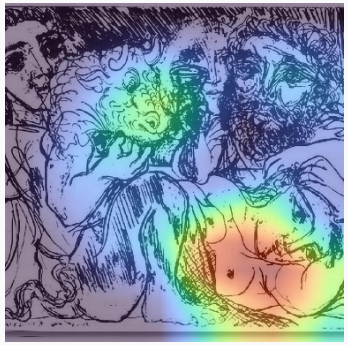

Excitement


Awe


Fear


Anger


Sadness


Disgust

Gradient-based attribution techniques come in many flavours, see this GitHub repository for an overview. Another approach to estimating saliency maps is occlusion techniques (Zeiler and Fergus 2014), which are precise but computationally expensive.

MC Dropout (Gal and Ghahramani 2016) approx. the predictive distribution using multiple stochastic forward passes[1]

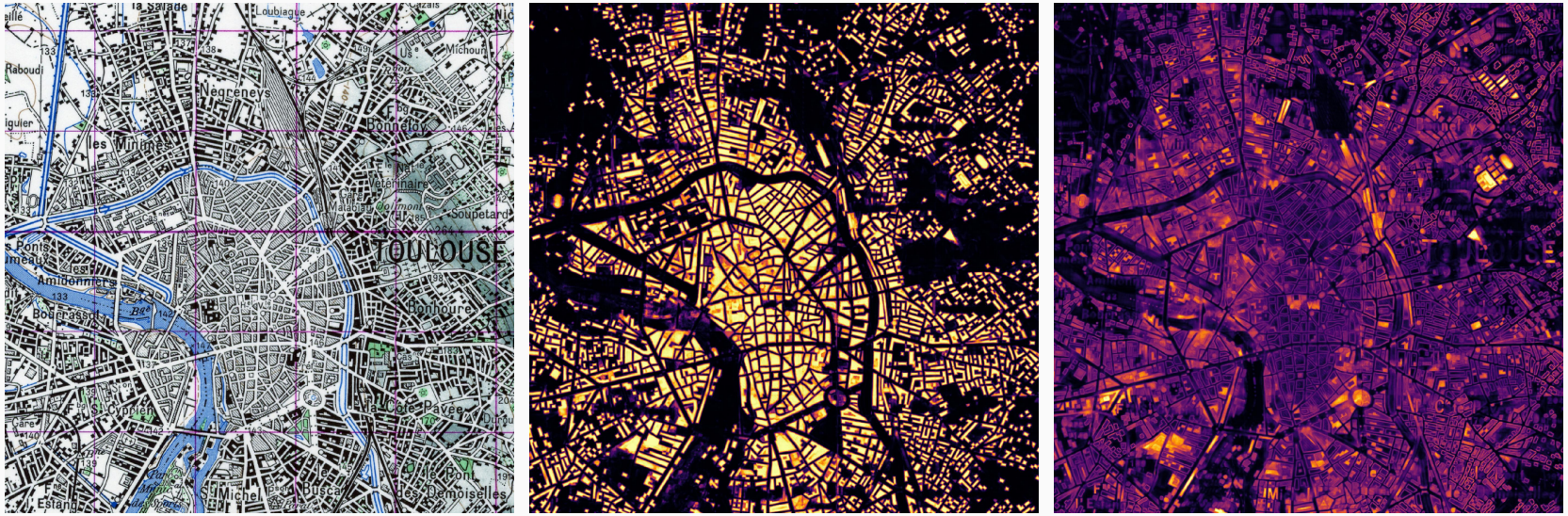
$$E[\hat{y}] \approx \frac{1}{T}\sum_{t=1}^{T} f^{(t)}(x) \quad \text{Var}[\hat{y}] \approx \frac{1}{T}\sum_{t=1}^{T} f^{(t)}(x)^2 - \left(\frac{1}{T}\sum_{t=1}^{T} f^{(t)}(x)\right)^2$$

where $t = 1 \ldots, T$ is the index of a stochastic forward pass

- Dropout layers remain active during inference to sample from the model's predictive distribution
- Each forward pass gives a different realisation of the model $\rightarrow$ Monte Carlo integration estimates uncertainty

[1] Approximate Bayesian variational inference

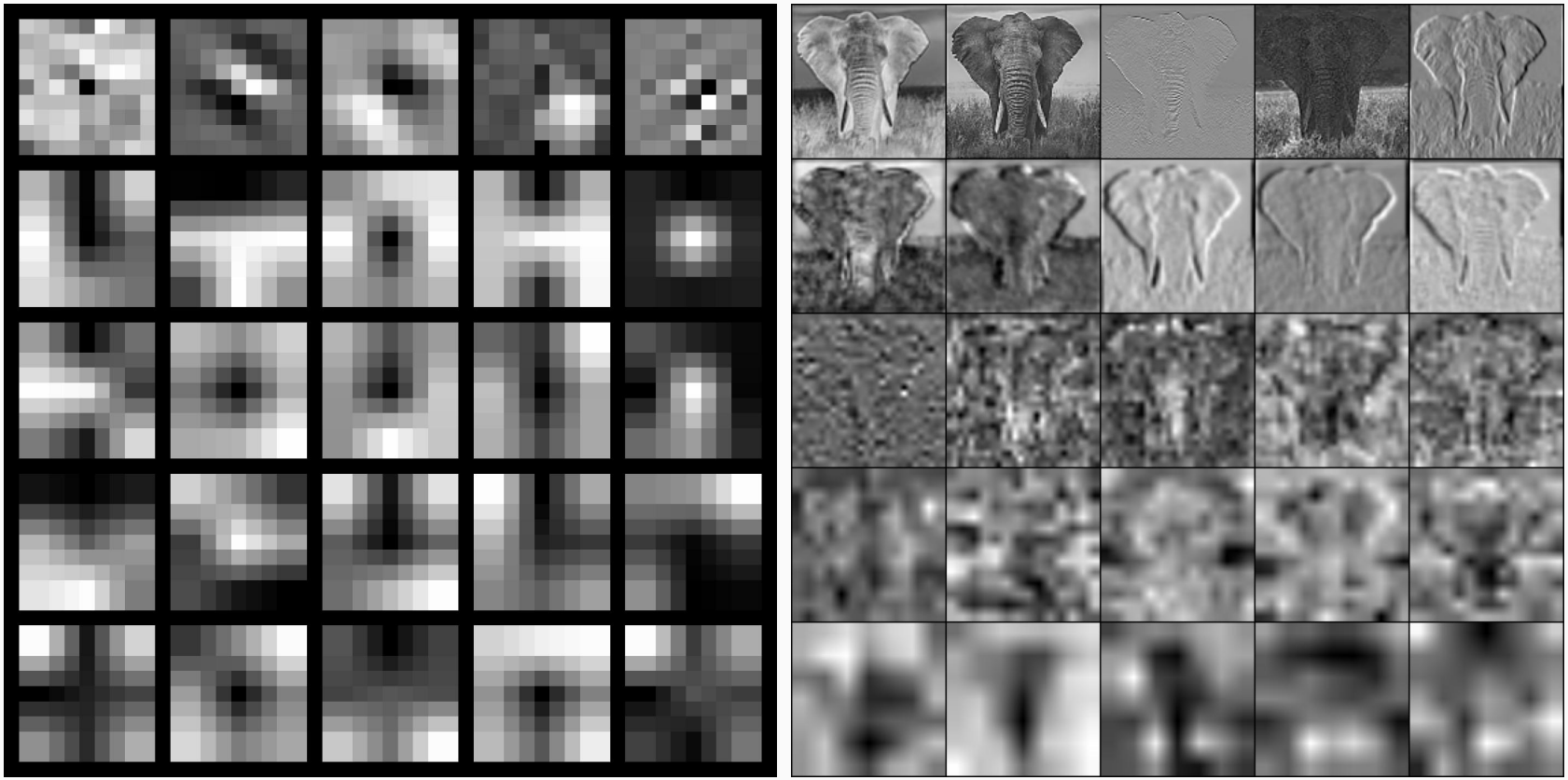
Building segmentation from historical maps



x_i $E[\hat{y}_i]$ $Var[\hat{y}_i]$

MC Dropout provides relative uncertainty estimates and requires calibration to align with true outcome probabilities.

Filters and activations



Filters are averaged along the d dimension. Images are normalised and resized to a common dimension.

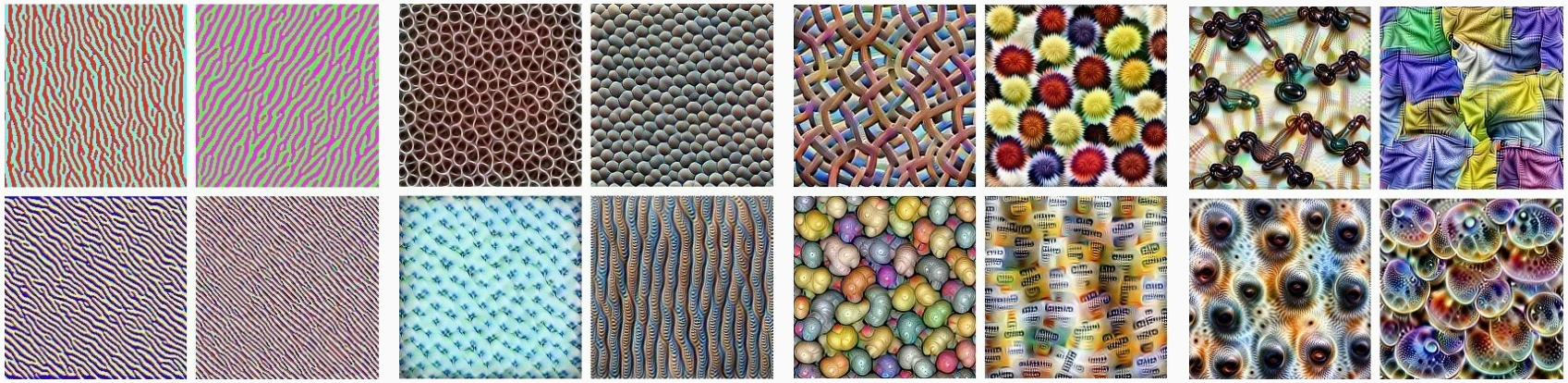
Filter visualisation (Zeiler and Fergus 2014) compute the input image that maximally activate specific filters

$$x^* = \underset{x}{argmax}\ A^d(x) - \lambda r(x)$$

$$A^d(x) = \frac{1}{hw} \sum_{i,j} A^d_{ij}(x)$$

where $A^d(x)$ is the activation of filter d given input x, averaged over spatial locations and $r(x)$ is a regularisation term. x is randomly initialised

Filter visualisation



Source: Olah (2017) for the GoogLeNet model (Szegedy et al. 2015)

- **Specialisation**: different groups of units specialise in detecting distinct features i.e. distributed representation
- **Hierarchy**: Multiple convolutions capture features at increasingly large spatial scales i.e. receptive field

We can optimise an existing image rather than random noise, which maginfies patterns detected by the model

Feature projections

Source: Mordvintsev et al. (2015) for the GoogLeNet model

Each feature map's scores indicate the presence, absence, and strength of a feature across spatial locations

Observations are mapped to numerical spaces where distances between pairs of points are well defined (Bengio et al. 2013)

Content space

Style space

Two-dimensional UMAP (McInnes et al. 2018) structure-preserving projection of 512-dimensional image embeddings from separate content and style encoders. Another popular projection preserving local distances is T-SNE (Maaten and Hinton 2008).

These representations disentangle the underlying factors of variation in the data (Wang et al. 2020)

Content neighbourhood

Style neighbourhood

Meaningful mathematical operations can be performed directly on these vectors, including distances (i.e. inner product), averaging, as well as positive or negative associations (addition, subtraction).

Optimisation

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

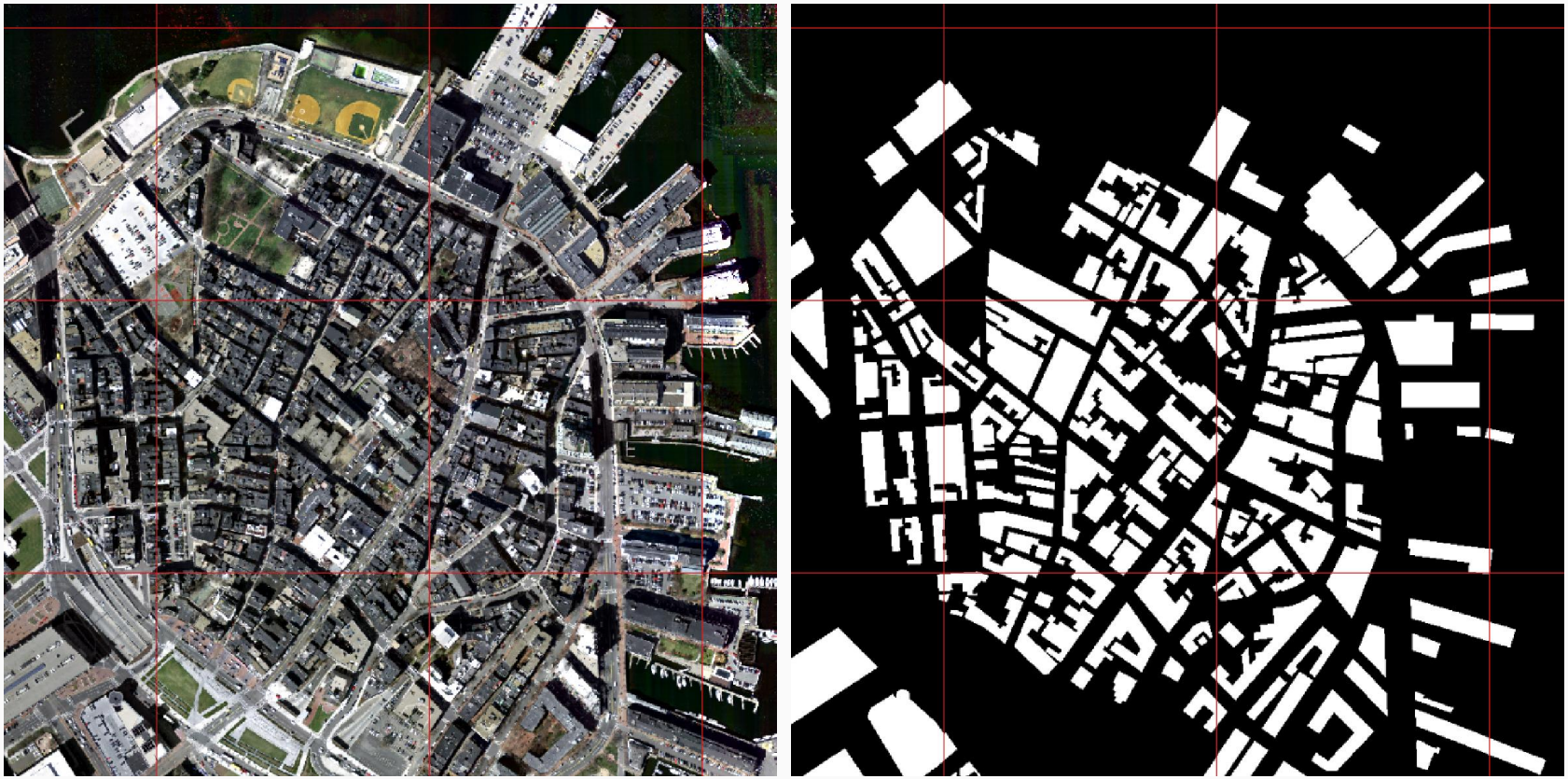
- Since the network can hardly rely on a particular input, it distributes more evenly the parameter values
- Dropout is not adapted to image data since features are captured using multiple neighbouring units
- Spatial dropout (Tompson et al. 2015) randomly keeps entire representations along the $d^{(l)}$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

Augmentation artificially increases the training sample, and enables the model to generalise better

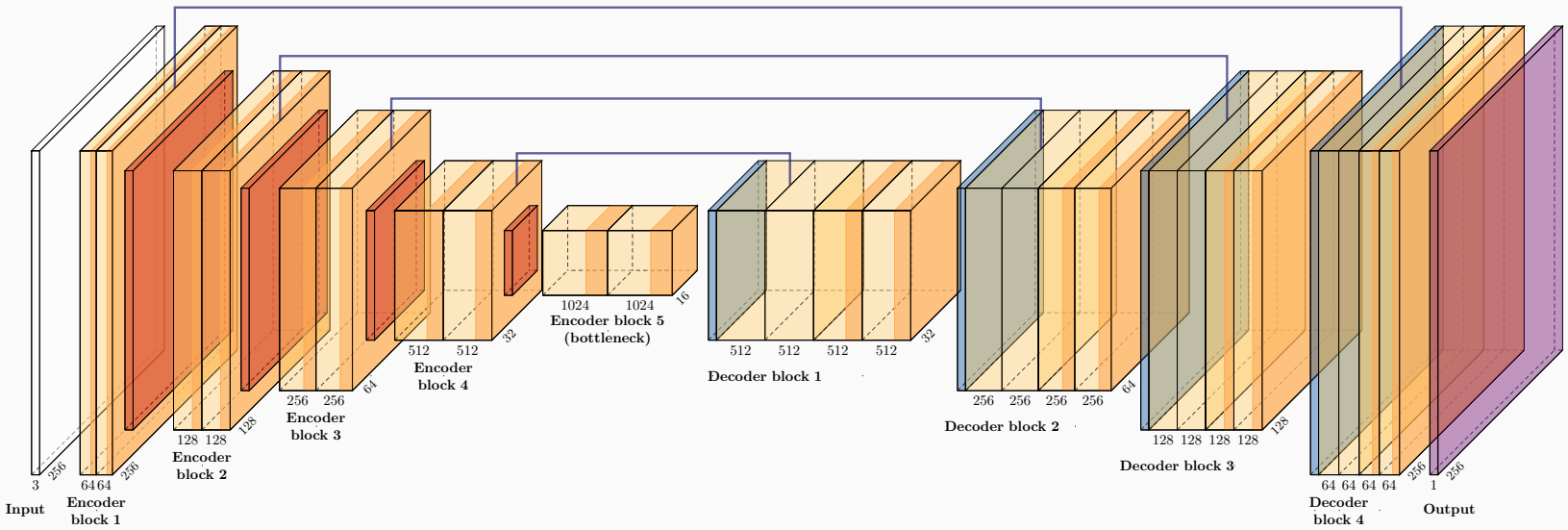
- Random transformations are applied to each image in the batch before each optimisation iteration
- Common operations involve flips, and parametrised shifts, rotations, zooms and changes in brightness
- Parameters are drawn from a distribution of “reasonable” values, as the transformations must look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

Segmentation

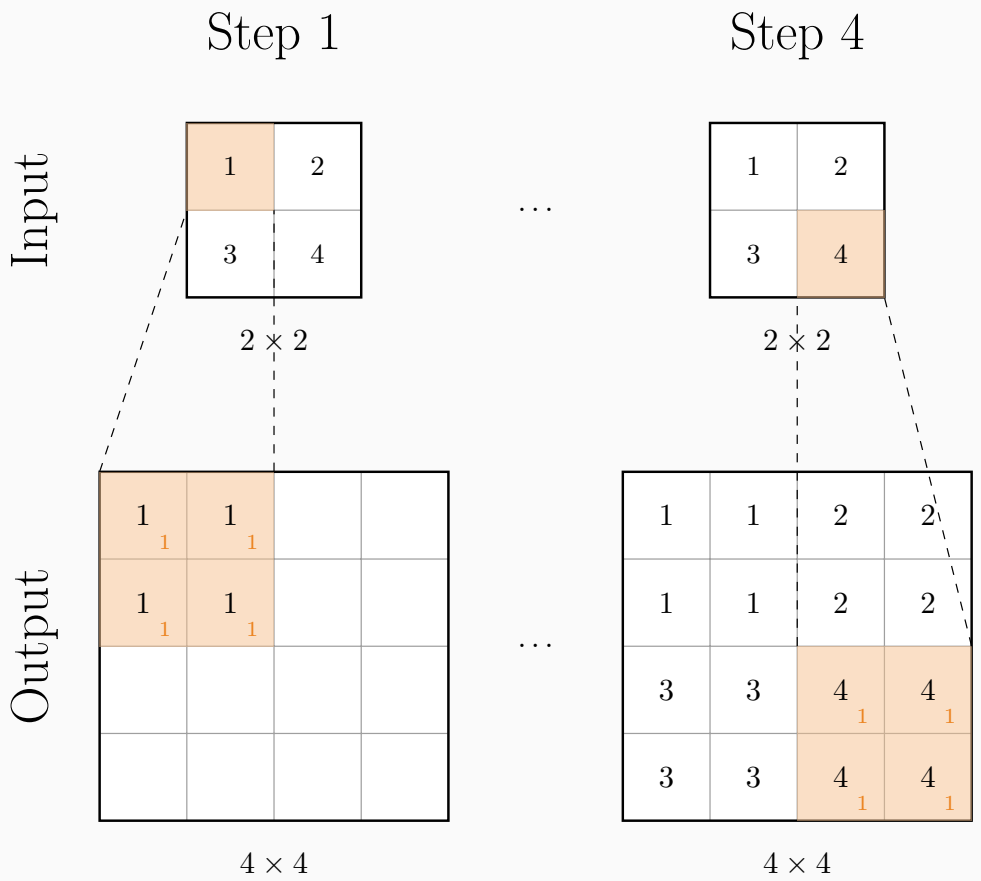
Segmentation of buildings from aerial images



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.



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 unit stride and zero padding.



Detailed 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 in involves many zero multiplications

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

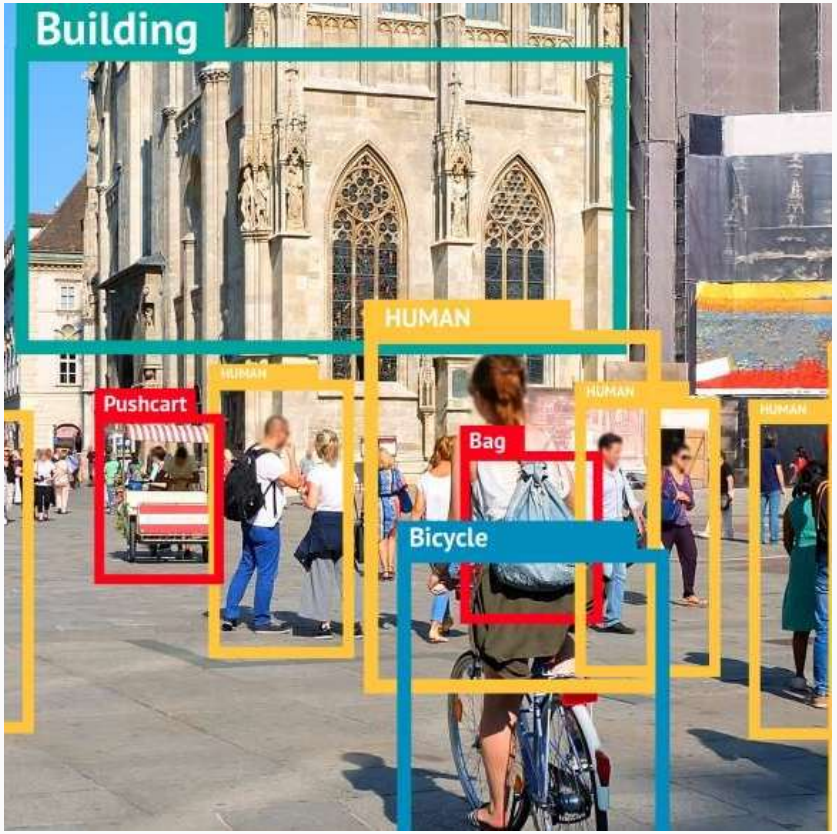
- The tensors are usually combined using concatenation (e.g. U-Net) or addition (e.g. ResNet)
- Improves optimisation by mitigating the vanishing gradients problem i.e. alternative for backpropagation
- Adds an identity mapping to the output of a layer, which can help preserve important features

Localisation

Localisation

Object localisation involves detecting, localising and classifying multiple objects in an image

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

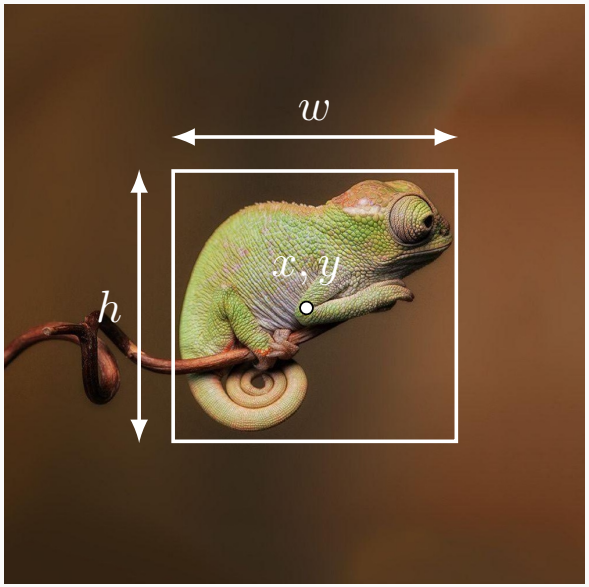


We study the YOLO V1 (You Only Look Once) model for object detection (Redmon et al. 2016)

- Convolutional network integrates multiple object localisation 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 computes robust representations and generalises to other domains e.g. paintings

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

$(0, 0)$



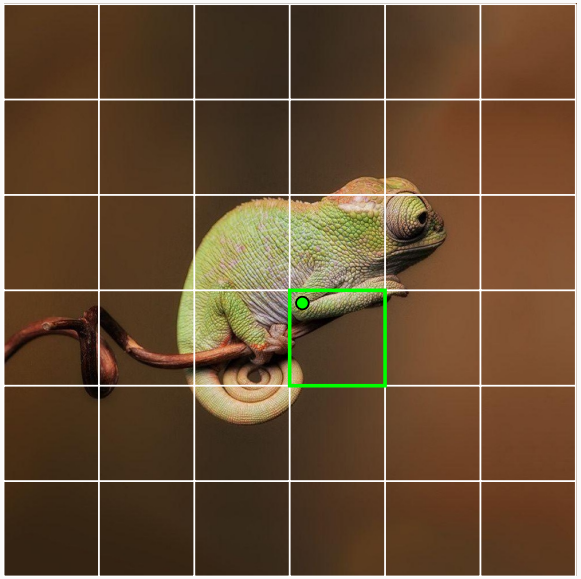
$(1, 1)$

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

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

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

- When $C < T$ (threshold), the class and localisation predictions are not meaningful
- For labels, objects spanning multiple cells are attributed to that containing its centre point

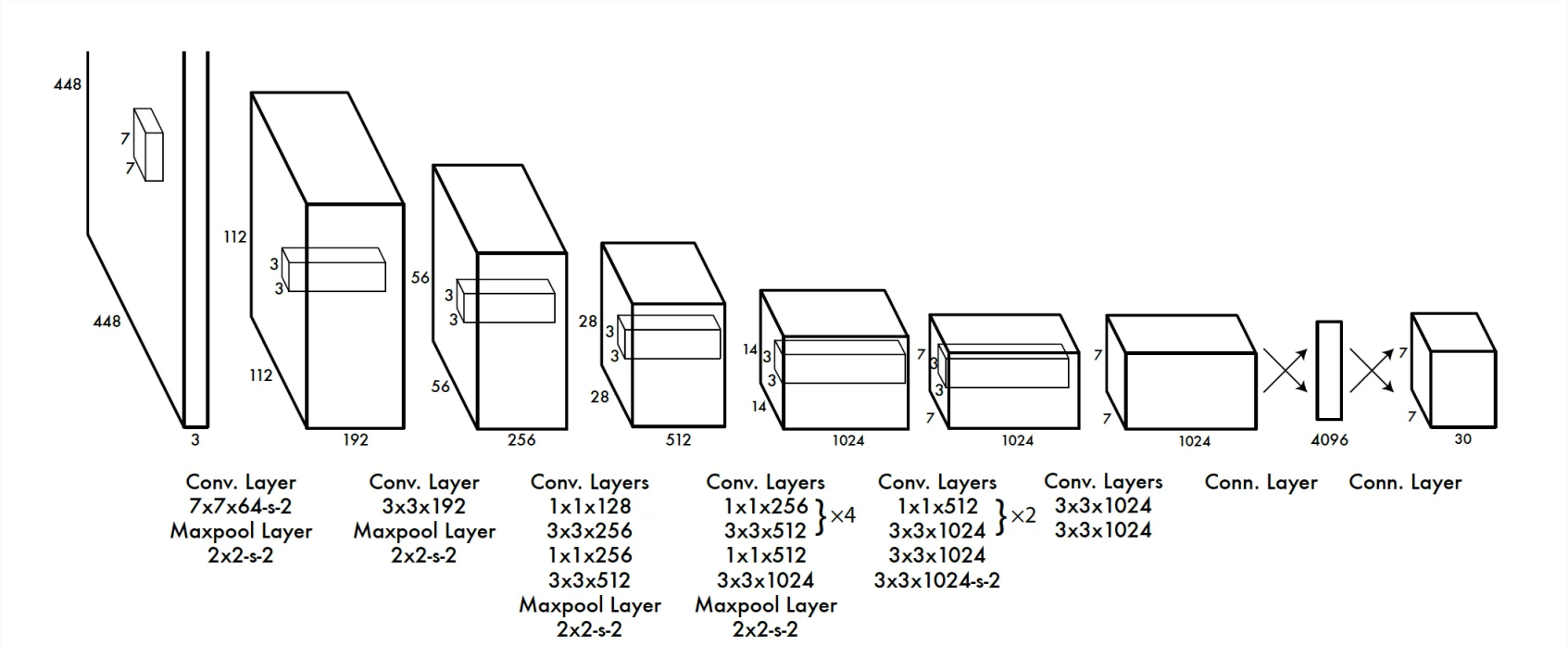


A common issue is that a single cell can contain multiple objects[2], potentially with different aspect ratios

- 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 specialisation between the bounding box predictors (e.g. sizes, aspect ratios, classes)

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

YOLO V1 architecture (Redmon et al. 2016)



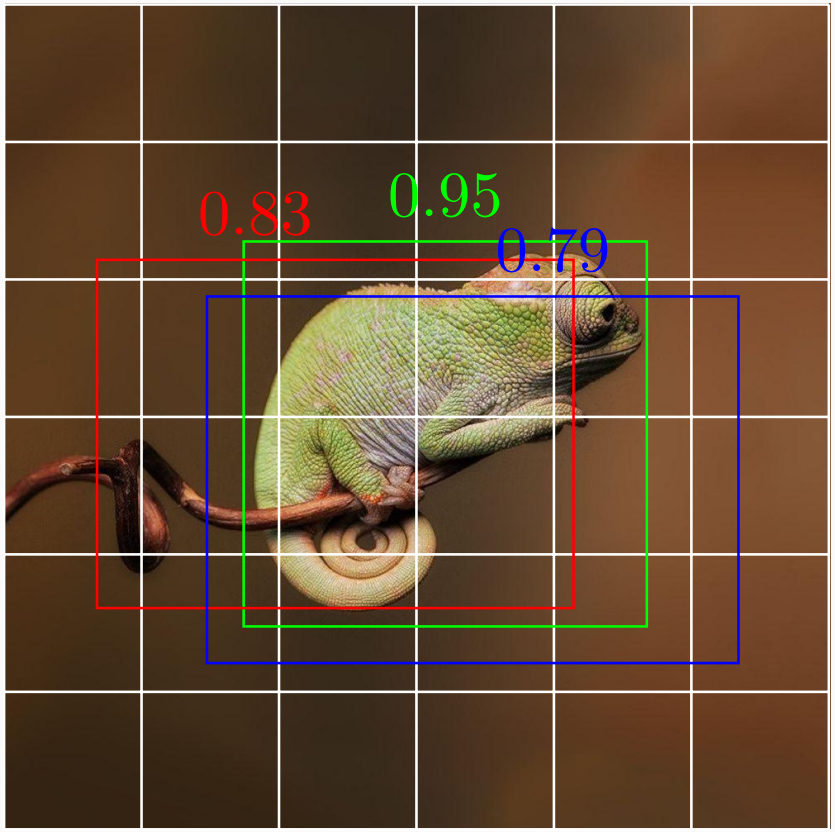
See Szegedy et al. (2015) and Lin et al. (2013) for the design of the convolutional layers. The Pascal VOC dataset (Everingham et al. 2015) has 20 classes and the model uses $S = 7$ and $B = 2$. The output tensor has dimensions $7 \times 7 \times (5 * 2 + 20)$. Each cell can only detect a single object.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

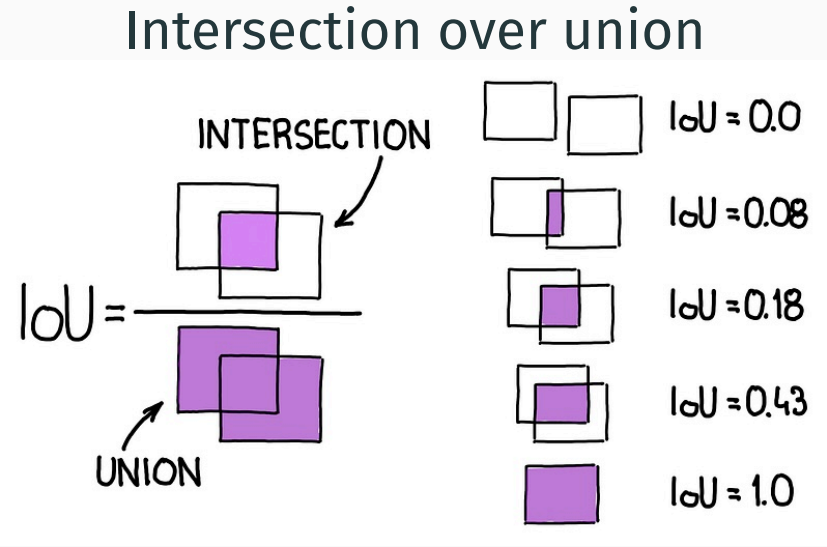
The YOLO loss function balances localisation, confidence, and class losses (but sum-squared errors are not always sensible)

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

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 $\lambda_{coord} = 5$ increases the importance of bounding box predictions and $\lambda_{noobj} = .5$ decreases that of confidence predictions for empty boxes. The squared root is used so that small deviations in large boxes matter less than in small boxes.

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





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

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 $IoU(b, r) > \tau$
3. Iterate until all boxes have been processed

Summary

Summary

- Image are represented as multidimensional arrays of pixel intensities with spatial and colour information
- Objects in images are represented by spatial arrangement of pixels with specific intensities
- Image models must learn features, or interaction patterns between pixels at various spatial scales
- Convolutional layers use discrete convolutions to capture interactions patterns across space and channels

Summary

- A convolutional network contains multiple convolutional layers, each using multiple convolutions, which gives rise to specialisation and hierarchy in the features
- Image modelling tasks include classification, segmentation (e.g. U-Net), localisation (e.g. YOLO) among others

Thank you for your attention!

References

References i

- 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. 17).
- Maaten, Laurens van der and Geoffrey Hinton (2008). **“Visualizing data using t-SNE”.** In: *Journal of Machine Learning Research* 9, pp. 2579–2606 (cit. on p. 31).
- LeCun, Yann, Corinna Cortes, and Christopher C.J. Burges (2010). **“MNIST handwritten digit database”.** In: *ATT Labs* (cit. on p. 5).

References ii

- 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. 31).
- Lin, Min, Qiang Chen, and Shuicheng Yan (2013). **“Network In Network”.** In: *CoRR* abs/1312.4400 (cit. on p. 47).
- 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. 34).
- Zeiler, Matthew D. and Rob Fergus (2014). **“Visualizing and understanding convolutional networks”.** In: *Computer Vision - ECCV 2014*, pp. 818–833 (cit. on pp. 24, 28).

References iii

- 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. 47).
- Mordvintsev, Alexander, Chris Olah, and Mike Tyka (2015). ***Inceptionism: Going Deeper into Neural Networks.*** Google Research Blog (cit. on p. 30).
- Nielsen, Michael A. (2015). ***Neural networks and deep learning.*** Determination Press.

References iv

- 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. 38).
- 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. 29, 47).
- 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. 34).

References v

- Dumoulin, Vincent and Francesco Visin (2016). **“A guide to convolution arithmetic for deep learning”.** In: *ArXiv e-prints* (cit. on p. 67).
- Gal, Yarin and Zoubin Ghahramani (2016). **“Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning”.** In: *International Conference on Machine Learning (ICML)*, pp. 1050–1059 (cit. on p. 25).
- Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). ***Deep Learning*.** MIT Press.

References vi

- 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. 43, 47).
- 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. 29).

References vii

- Selvaraju, Ramprasaath R et al. (2017). **“Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization”.** In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 618–626 (cit. on p. 22).
- Wu, Jianxin (2017). **“Introduction to convolutional neural networks”.** Lecture notes.
- McInnes, Leland et al. (2018). **“UMAP: Uniform Manifold Approximation and Projection”.** In: *Journal of Open Source Software* 3.29, p. 861 (cit. on p. 31).
- 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. 32).

References viii

- Zhang, Aston et al. (2023). ***Dive into deep learning***. Cambridge University Press.

Appendix

Appendix

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

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

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

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

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

◂ Presentation

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

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

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

- Padding helps preserving 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)

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 (parametrised) 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

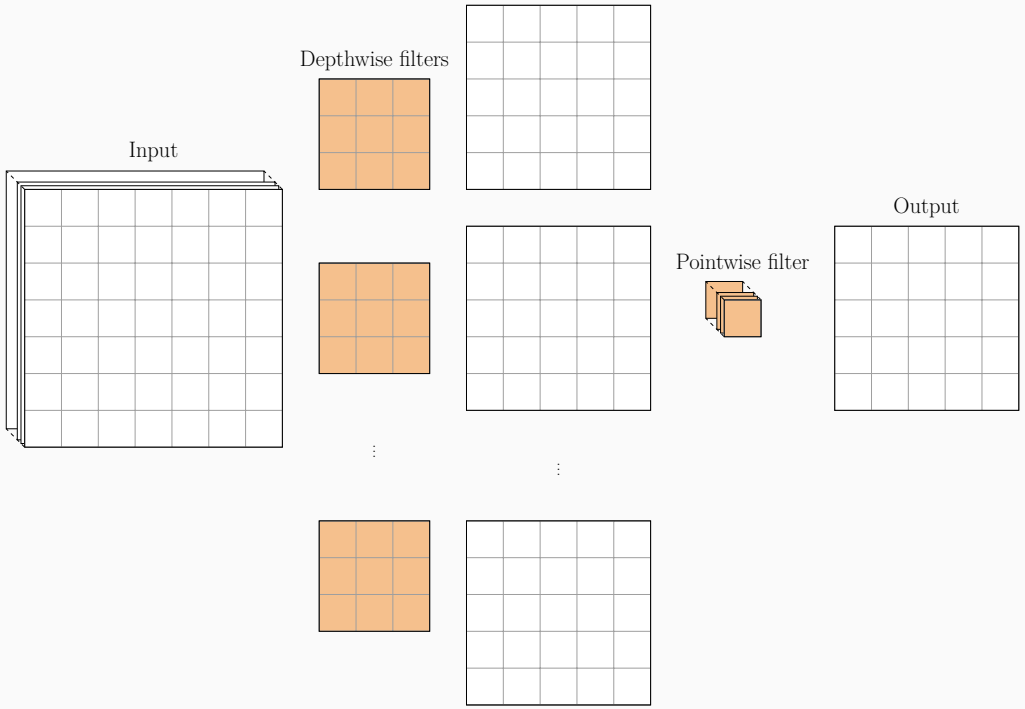
Consider the following convolution operation

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

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

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

Depthwise separable convolutions



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.

◂ Presentation

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 $(H^{(l)} - h_\beta + 1) \times (W^{(l)} - w_\beta + 1) \times D$

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

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