AI Research Labs

Deep Learning Foundations · Practice test 10 · 20 questions

Convolutional Neural Networks

How to use this test. Answer all 20 questions without notes, then mark yourself with the answer key at the back. Each question has exactly one best answer. Score 75% or more and you've mastered the material; below that, reread the lessons for the questions you missed.

10.1 From Fully Connected to Convolutional

Q1 `calculate` A 32×32×3 image is flattened and fully connected to 100 units. How many parameters does that layer have?

- A 102,400
- B 307,200
- C 3,172
- D 307,300

Q2 `calculate` A convolutional layer has 16 filters of size 3×3 over a 3-channel image. How many parameters does it have?

- A 49,152
- B 432
- C 160
- D 448

Q3 `concept` What does weight sharing mean in a convolutional layer?

- A Every filter in the layer has identical weight values
- B The same filter weights are used at every position
- C Weights are copied between the layers of the network
- D Each output unit connects to every input value

Q4 `interpret` Why does the conv layer's parameter count not grow when the input image gets larger?

- A Its parameters are recomputed for each new image
- B Convolutions ignore every pixel outside the centre
- C It depends on filter size and count, not positions
- D Larger images are resized down before the conv layer

Q5 `concept` Which two properties of images do convolutional layers exploit?

- A Locality and translation of patterns
- B Colour and the brightness of the pixels
- C The labels and their class frequencies
- D Image size and the file format used

10.2 The Convolution Operation

Q6 calculate Input size 4×4, filter 3×3, stride 1, no padding. What size is the feature map?

A 3×3
B 4×4
C 1×1
D 2×2

Q7 calculate A 3×3 patch has column sums 3, 4 and 3 (left to right). The filter weights its columns $+1, 0, -1$. What is the output?

A −6
B 10
C 6
D 0

Q8 calculate Input size 32, filter size 5, padding 2, stride 1. What is the output size?

A 30
B 28
C 32
D 16

Q9 concept What does the filter with columns $(+1, 0, -1)$ respond to most strongly?

A A single bright pixel in the centre
B A uniformly bright patch of pixels
C A horizontal edge, brighter at the top
D A vertical edge, brighter on the left

Q10 concept A layer has 16 filters applied to a 3-channel image. How many feature maps does it produce?

A 16, one per filter
B 48, filters × channels
C 3, one per channel
D 1, summed over filters

10.3 Pooling

Q11 calculate Max pooling (2×2) is applied to the window $\{1, 3, 4, 2\}$. What is the output?

A 1
B 2.5
C 4
D 10

Q12 `calculate` Average pooling (2×2) is applied to the window $\{0, 1, 2, 6\}$. What is the output?

- A 6
- B 1.5
- C 2.25
- D 9

Q13 `calculate` 2×2 pooling with stride 2 is applied to a 16×16 feature map. What size is the output?

- A 15×15
- B 14×14
- C 4×4
- D 8×8

Q14 `concept` How many learned parameters does a max-pooling layer have?

- A One per input channel
- B Four per 2×2 window
- C One weight per window
- D None

Q15 `interpret` A feature shifts by one pixel within a 2×2 pooling window. Why does max pooling's output often stay the same?

- A Pooling layers retrain their weights after each shift
- B The convolution before it ignores all small shifts
- C The maximum within the window doesn't change
- D Pooling averages the shift over the whole image

10.4 CNN Components and Feature Hierarchies

Q16 `concept` What is the role of the fully connected layers at the end of a CNN?

- A Detect edges and corners in the raw input image
- B Add the non-linearity between convolution layers
- C Shrink the feature maps to reduce the computation
- D Combine the extracted features into a prediction

Q17 `concept` In a trained CNN, what do the early layers typically detect?

- A Whole objects such as faces and cars
- B Edges, corners and colour blobs
- C Textures and object parts like wheels
- D The final class label of the image

Q18 `apply` Why put a ReLU after each convolution?

- A To normalise the input image into the range from 0 to 1
- B So stacked convolutions don't collapse into one linear map
- C To share the filter weights across different positions
- D To shrink the spatial size of the resulting feature maps

Q19 `interpret` Why does the edge → part → object hierarchy help CNNs generalise on images?

- A It removes the need for any labelled training data at all
- B It guarantees the same accuracy on every image dataset
- C It matches how images are composed, so features are reused
- D It lets the network memorise each training image exactly

Q20 `concept` A unit in the second 3×3 conv layer sees a 3×3 patch of layer-1 maps. How large a region of the original image does it depend on?

- A 9×9
- B 5×5
- C 3×3
- D 6×6

Answer key

Each answer comes with the reasoning, including why the most tempting wrong option is wrong.

Q1 **D** · lesson 10.1

$32 \times 32 \times 3 = 3{,}072$ inputs, so $3{,}072 \times 100 + 100 = 307{,}300$. 307,200 forgets the biases, 3,172 adds instead of multiplying, and 102,400 ignores the 3 colour channels.

Q2 **D** · lesson 10.1

Each filter has $3 \times 3 \times 3 = 27$ weights plus a bias: $16 \times 27 + 16 = 448$. 432 forgets the biases, 160 ignores the channels, and 49,152 counts one filter copy per pixel, which weight sharing avoids.

Q3 **B** · lesson 10.1

One filter slides across the whole input with the same weights, so a pattern is detected wherever it appears and the parameter count doesn't depend on the image size. Different filters have different weights; full connectivity is the FC layer.

Q4 **C** · lesson 10.1

Thanks to weight sharing, a 3×3×3 filter has 27 weights whether it slides over a 32×32 or a 1000×1000 image; only the feature map gets bigger. The image isn't resized and every pixel is covered.

Q5 **A** · lesson 10.1

Nearby pixels are related (local connectivity) and the same pattern can appear anywhere (weight sharing). Colour and brightness are inputs the filters process, not the structural ideas behind the layer.

Q6 **D** · lesson 10.2

$\lfloor (n + 2p - k)/s \rfloor + 1 = (4 - 3)/1 + 1 = 2$. 4×4 would need padding 1, and 1×1 would need a 4×4 filter. The filter fits in 2 positions along each axis.

Q7 **D** · lesson 10.2

$3 \times 1 + 4 \times 0 + 3 \times (-1) = 0$: no left-to-right change in brightness, so no vertical edge. 10 just adds the sums, 6 forgets the minus sign on the right column, and −6 flips it.

Q8 **C** · lesson 10.2

$(32 + 2 \times 2 - 5)/1 + 1 = 32$: padding of 2 keeps the size unchanged for a 5×5 filter. 28 ignores the padding, and 16 would need stride 2.

Q9 **D** · lesson 10.2

It subtracts the right column from the left, so it's large when the left side is brighter: a vertical edge. A uniform patch gives 0, and a horizontal edge needs row weights such as $(+1, 0, -1)$ down the rows.

Q10 **A** · lesson 10.2

Each filter spans all 3 input channels and sums over them, producing one feature map. 16 filters give 16 maps, which become 16 input channels for the next layer.

Q11 **C** · lesson 10.3

Max pooling keeps the largest value, 4. 2.5 would be average pooling, 10 the sum, and 1 the minimum.

Q12 **C** · lesson 10.3

$(0 + 1 + 2 + 6)/4 = 9/4 = 2.25$. 6 is the max, 9 the sum, and 1.5 averages only the middle two values.

Q13 **D** · lesson 10.3

Non-overlapping 2×2 windows halve each dimension: 16/2 = 8. 15×15 would be stride 1, and 4×4 would be pooling twice.

Q14 **D** · lesson 10.3

Pooling applies a fixed rule (max or average) with no weights or biases to learn. It just summarises each window.

Q15 **C** · lesson 10.3

If the strong response moves but stays inside the same window, the max is unchanged, which gives a degree of translation invariance. Pooling has no weights, and the convolution's output does move with the feature.

Q16 **D** · lesson 10.4

After convolution and pooling extract features, dense layers (with a softmax for classification) turn them into the final output. Edges are detected by early conv layers, pooling shrinks maps, and activations add non-linearity.

Q17 **B** · lesson 10.4

Early filters respond to simple local patterns; middle layers combine them into textures and parts; deep layers respond to whole objects. Class labels come from the final dense layers.

Q18 **B** · lesson 10.4

Convolution is linear, so without a non-linearity a stack of conv layers is equivalent to one. ReLU lets depth build richer features. Shrinking is pooling's job; weight sharing is built into convolution.

Q19 **C** · lesson 10.4

Objects are made of parts, and parts of edges; learning features that mirror that structure means the same detectors serve many images and classes. That's why hand-crafted features are no longer needed, not because labels or memorisation are.

Q20 **B** · lesson 10.4

Each layer-1 unit covers 3×3 pixels, and 3×3 neighbouring layer-1 units together cover $3 + 2 = 5$ pixels in each direction. This receptive field grows with depth (and with pooling). 9×9 would multiply instead of adding.