AI Research Labs

Deep Learning Foundations · Practice test 4 · 24 questions

Activation Functions

How to use this test. Answer all 24 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.

4.1 Sigmoid, tanh, ReLU and Friends

Q1 `calculate` A sigmoid unit outputs $a = 0.8$. What is its derivative $\sigma'(z)$ at that point?

A. 0.16
B. 0.8
C. 0.2
D. 0.64

Q2 `concept` What is the largest value the sigmoid derivative can take, and where?

A. 0.50, at $z = 0$
B. 1.00, as $z \to \infty$
C. 1.00, at $z = 0$
D. 0.25, at $z = 0$

Q3 `concept` Why can't networks of sign-activation units be trained by gradient descent?

A. Its derivative is zero almost everywhere, so there's no signal
B. It is too expensive to compute for large networks of units
C. Its derivative is infinite everywhere except exactly at zero
D. Its outputs are only −1 and +1, which the loss can't measure

Q4 `interpret` A hidden unit has pre-activation $z = 5$. Compared with $z = 0$, how much gradient does a sigmoid pass through it?

A. Far more: about 0.99 instead of 0.5
B. The same, since the derivative is constant
C. Exactly half as much, because the output doubled
D. Far less: about 0.0066 instead of 0.25

Q5 `concept` What advantage does tanh have over the sigmoid for hidden units?

A. It never saturates, whatever the size of the input
B. Its derivative is exactly 1 everywhere it is defined
C. It is zero-centred, with outputs in (−1, 1)
D. It produces exact zeros for every negative input

4.2 Vanishing Gradients

Q6 `calculate` Backpropagating through 10 sigmoid layers multiplies 10 activation derivatives. What is the largest that product can be?

- A About 0.025
- B 1
- C About 9.5×10^{-7}
- D 2.5

Q7 `interpret` In a deep sigmoid network, the last layers' weights change quickly but the first layers' barely move. What explains it?

- A The first layers' inputs are always zero in deep networks of this kind
- B The first layers have a much smaller learning rate set by the optimiser
- C The error signal shrinks at every layer it passes back through
- D The loss function only depends on the weights of the last layer

Q8 `apply` Which change most directly reduces vanishing gradients in a deep network?

- A Replace the sigmoid hidden units with ReLU
- B Use a much smaller learning rate throughout
- C Add a sigmoid output layer for the predictions
- D Make the input features larger in magnitude

Q9 `concept` How does saturation make vanishing gradients worse?

- A Saturation makes the learning rate negative
- B Saturated units have derivatives far below 0.25
- C Saturated units output exactly zero every time
- D Saturated units double the error signal each time

Q10 `concept` In the recursion $\delta^{(k)} = D_k W_{k+1}^{\top} \delta^{(k+1)}$, what does D_k contain?

- A The learning rates for each unit on its diagonal
- B The weights of layer k on its diagonal
- C The inputs to layer k on its diagonal
- D The activation derivatives $f'(z^{(k)})$ on its diagonal

4.3 Dying ReLU, Leaky ReLU and ELU

Q11 `concept` What is a "dead" ReLU unit?

- A One whose weights have become very large after a big update step
- B One whose input is negative for every example, so it never updates
- C One whose output is always exactly 1, so it passes everything through
- D One that was removed from the network by dropout during training

Q12 apply Many hidden units in your ReLU network output 0 for every input after a few epochs. Which fix targets this?

A Remove the biases from all the hidden layers
B Add many more ReLU layers to the network
C Use Leaky ReLU and a smaller learning rate
D Switch every hidden unit to the sign function

Q13 calculate A Leaky ReLU with $\alpha = 0.01$ receives $z = -3$. What does it output?

A −3
B −0.03
C 0
D 0.03

Q14 concept How do vanishing gradients and dying ReLU differ?

A Vanishing affects biases, while dying affects only the weights
B Vanishing hits ReLU networks, while dying affects only sigmoid ones
C Vanishing: small factors across layers. Dying: one unit stuck at 0.
D They are the same problem, only named for different activations

Q15 interpret Why is ReLU still the default hidden activation despite dying units?

A Because it never saturates for any input, positive or negative
B Because it guarantees that no unit can ever stop learning at all
C Its derivative is 1 when active, and it's cheap and sparse
D Because its output is bounded, which keeps all activations small

4.4 Choosing Activation Functions

Q16 apply A network predicts tomorrow's temperature in °C. Which output activation fits?

A Softmax
B Sigmoid
C ReLU
D Linear

Q17 apply A network decides whether an email is spam. Which output layer fits?

A One sigmoid unit giving $P(\text{spam})$
B One linear unit giving an unbounded score
C Ten softmax units, one for each spam score
D One ReLU unit giving a non-negative score

Q18 apply A network recognises which of 10 digits is in an image. Which output layer fits?

A 1 unit with sigmoid
B 1 unit with linear output
C 10 units with ReLU
D 10 units with softmax

Q19 concept For the hidden layers of a deep network, which activation is the recommended default?

A ReLU
B Linear
C Sign
D Sigmoid

Q20 interpret Why is a sigmoid acceptable in a network with one hidden layer but discouraged in a 20-layer one?

A Sigmoids become linear once there are more than two layers
B Vanishing gradients only become severe with many layers
C Sigmoids can only be used in networks with one layer
D Deep networks require outputs bounded in (−1, 1)

Q21 apply In the 2–3–1 network from lesson 3.1, why does the output neuron use the identity activation?

A Because output neurons have no weights
B So the output always lies between 0 and 1
C Because ReLU cannot be used in a last layer
D So the prediction can be any real number

Q22 concept Is a neuron with no activation function still a neuron?

A No: without an activation it is just a weight
B No: it becomes a bias term on the next layer
C Yes: it uses the identity activation
D Yes, but only if it is in the input layer

Q23 concept A neuron applies ReLU to its weighted sum, then a sigmoid to the result. How many activation functions does it really have?

A One: the two together form a single activation
B Two, and each one needs its own weights and bias
C Two, so it counts as two separate neurons
D None, because the two functions cancel out

Q24 concept Can a neuron contain another neuron inside it?

A Yes: hidden neurons contain the input neurons
B Yes, whenever it uses more than one activation
C Only in convolutional networks, where filters nest
D No: networks are built by connecting neurons

Answer key

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

Q1 **A** · lesson 4.1

Using the output directly, $\sigma' = a(1-a) = 0.8 \times 0.2 = 0.16$. The tempting 0.8 is the output itself, 0.2 is only $1-a$, and 0.64 is a^2.

Q2 **D** · lesson 4.1

$\sigma'(z) = \sigma(1-\sigma)$ peaks when $\sigma = 0.5$, i.e. $z = 0$, giving $0.5 \times 0.5 = 0.25$. A derivative of 1 at 0 belongs to tanh; 0.5 is the sigmoid's value there, not its slope.

Q3 **A** · lesson 4.1

Small weight changes don't change a sign unit's output unless they cross zero, so the derivative is 0 (and undefined at 0). Gradient descent gets no direction to move in. The loss can measure ±1 outputs fine, and the function is cheap.

Q4 **D** · lesson 4.1

At $z = 5$, $\sigma \approx 0.993$ and $\sigma' = 0.993 \times 0.007 \approx 0.0066$, almost 40 times smaller than 0.25. The unit is saturated. 0.99 is the output, not the derivative.

Q5 **C** · lesson 4.1

tanh is a rescaled sigmoid ($2\sigma(2z) - 1$) centred on zero, which often helps training. It still saturates for large $|z|$, its derivative $1 - \tanh^2$ is only 1 at zero, and exact zeros are a ReLU property.

Q6 **C** · lesson 4.2

Each factor is at most 0.25, so the product is at most $0.25^{10} = 4^{-10} \approx 9.5 \times 10^{-7}$. 2.5 adds the factors, 0.025 multiplies 0.25 by 0.1, and 1 would need every derivative to be 1 (as for active ReLUs).

Q7 **C** · lesson 4.2

Each layer multiplies the backpropagated signal by $\sigma' \leq 0.25$ (and the weights), so early layers receive tiny gradients: vanishing gradients. The learning rate is shared, inputs aren't zero, and the loss depends on every weight.

Q8 **A** · lesson 4.2

ReLU's derivative is 1 for active units, so it doesn't shrink the signal at each layer. Larger inputs push sigmoids into saturation (worse), and the learning rate scales updates without fixing the vanishing signal.

Q9 **B** · lesson 4.2

For large $|z|$ the sigmoid is nearly flat, so σ' is close to 0, much smaller than its maximum 0.25. Saturated sigmoids output values near 0 or 1, not exactly zero, and they shrink rather than amplify the signal.

Q10 **D** · lesson 4.2

$D_k = \mathrm{diag}(f'(z^{(k)}))$ multiplies the error by each unit's activation derivative. Unrolling the recursion gives one such factor per layer, which is the vanishing-gradient mechanism. Weights appear separately as $W^\top$.

Q11 **B** · lesson 4.3

If $z < 0$ for all training examples, the unit outputs 0 and its gradient is 0, so its incoming weights never change and it stays inactive. Dropout removes units only temporarily, and ReLU outputs aren't capped at 1.

Q12 **C** · lesson 4.3

Leaky ReLU keeps a small slope for negative inputs, so dead units can recover, and a smaller learning rate avoids the big steps that kill them. Sign units can't be trained at all, and more ReLU layers can add more dead units.

Q13 **B** · lesson 4.3

For $z < 0$ the output is $\alpha z = 0.01 \times (-3) = -0.03$. Plain ReLU would give 0; -3 would be the identity; the sign stays negative.

Q14 **C** · lesson 4.3

Vanishing gradients come from multiplying many derivatives below 1 (typical of sigmoids), so early layers learn slowly. Dying ReLU is a single unit with zero output and zero gradient. Different mechanisms, different remedies.

Q15 **C** · lesson 4.3

Active ReLUs pass gradients without shrinking them, the function is very cheap, and many units output exactly 0 (sparsity). It does "saturate" at 0 for negative inputs (that's dying), and its output is unbounded above.

Q16 **D** · lesson 4.4

A temperature can be any real number, including negative values, so the output must be unbounded: linear. A sigmoid caps it in (0, 1), softmax is for class probabilities, and ReLU can't output negative temperatures.

Q17 **A** · lesson 4.4

Binary classification needs one probability, which a sigmoid provides. A linear unit isn't a probability, ten softmax units would be a 10-class problem, and ReLU outputs aren't bounded by 1.

Q18 **D** · lesson 4.4

Softmax over 10 units gives one probability per digit, summing to 1. A single sigmoid handles only two classes, ReLU outputs don't form a distribution, and a single linear number would impose a false order on the digits.

Q19 **A** · lesson 4.4

ReLU avoids shrinking gradients for active units and is cheap. Sign can't be trained, sigmoid causes vanishing gradients in deep stacks (fine for shallow nets), and linear hidden layers collapse into one.

Q20 **B** · lesson 4.4

One or two factors of at most 0.25 are tolerable; twenty of them leave the early layers with almost no gradient. Sigmoids work at any depth in principle and stay non-linear; the issue is purely the gradient signal.

Q21 **D** · lesson 4.4

The network does regression, so its output must be free to take any value. ReLU can be used in a last layer, but it would forbid negative predictions; a range of 0 to 1 is what a sigmoid gives; and the output neuron has three weights and a bias of its own.

Q22 **C** · lesson 4.4

"No activation" means the identity, $f(z) = z$: the neuron still computes a weighted sum plus a bias and passes it on. Regression output neurons are like this, and linear regression is exactly one such neuron. Input-layer units hold the data and compute nothing, so they aren't the example.

Q23 **A** · lesson 4.4

A neuron computes one weighted sum and applies one function to it. Applying two functions in a row is just one composite function, $\sigma(\mathrm{ReLU}(z))$, with no extra weights. It doesn't become two neurons, and the functions don't cancel: the result lies between 0.5 and 1.

Q24 **D** · lesson 4.4

A neuron is the smallest unit: weights, a bias and one activation. Structure comes from connections, with each layer's outputs feeding the next layer. Hidden neurons receive the inputs; they don't contain them. Convolutional filters are shared weights, not neurons inside neurons.