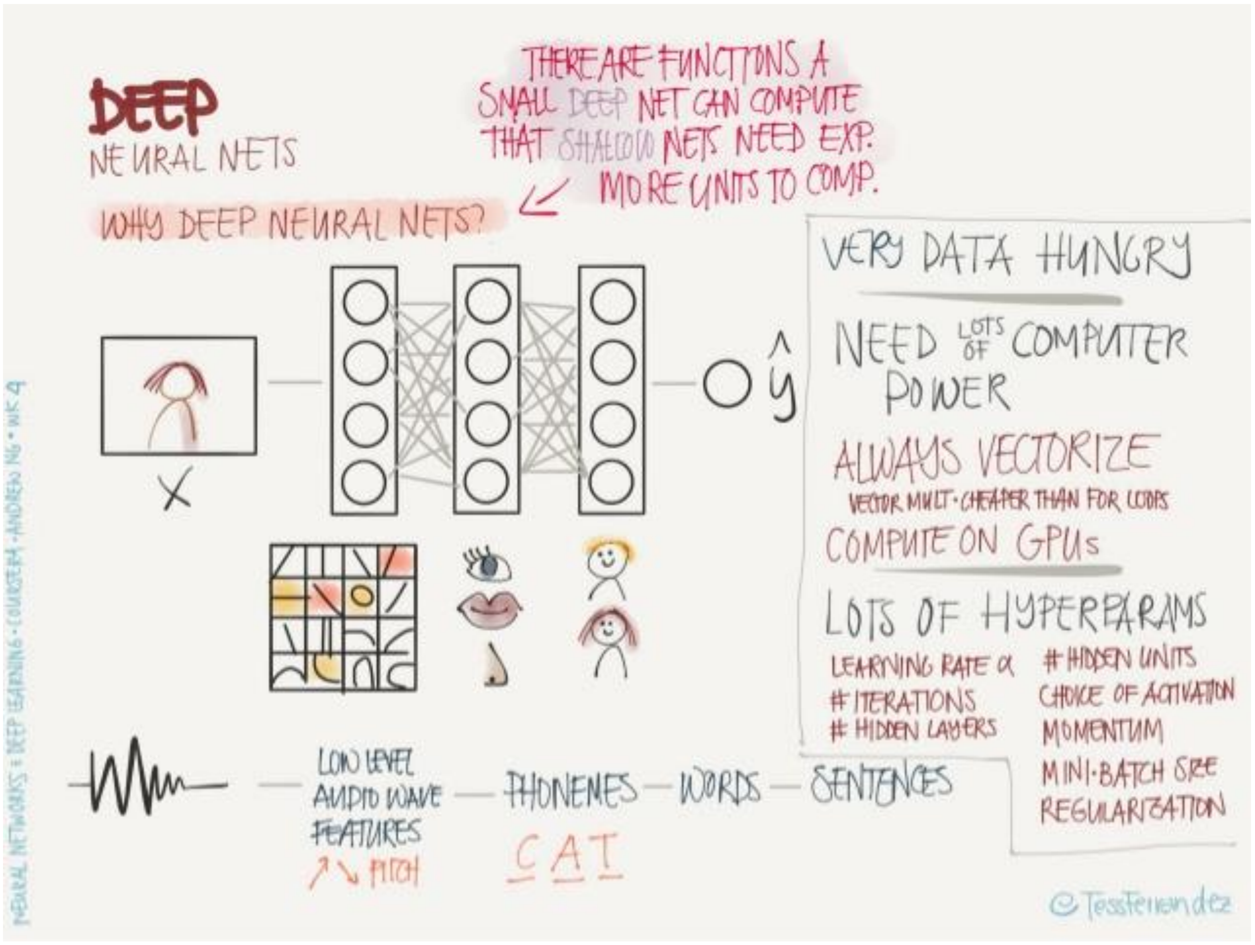


Deep Neural Networks

Understand the key computations underlying deep learning, use them to build and train deep neural networks, and apply it to computer vision.

Deep L-layer neural network

- Shallow NN is a NN with one or two layers.
- Deep NN is a NN with three or more layers.
- We will use the notation `L` to denote the number of layers in a NN.
- `n[l]` is the number of neurons in a specific layer `l`.
- `n[0]` denotes the number of neurons input layer. `n[L]` denotes the number of neurons in output layer.
- `g[l]` is the activation function.
- `a[l] = g[l](z[l])`
- `w[l]` weights is used for `z[l]`
- `x = a[0]`, `a[l] = y'`
- These were the notation we will use for deep neural network.
- So we have:
  - A vector `n` of shape `(1, NoOfLayers+1)`
  - A vector `g` of shape `(1, NoOfLayers)`
  - A list of different shapes `w` based on the number of neurons on the previous and the current layer.
  - A list of different shapes `b` based on the number of neurons on the current layer.

Forward Propagation in a Deep Network

- Forward propagation general rule for one input:
- `z[l] = W[l]a[l-1] + b[l]`
- `a[l] = g[l](a[l])`
- Forward propagation general rule for `m` inputs:
- `Z[l] = W[l]A[l-1] + B[l]`
- `A[l] = g[l](A[l])`
- We can't compute the whole layers forward propagation without a for loop so its OK to have a for loop here.
- The dimensions of the matrices are so important you need to figure it out.

Getting your matrix dimensions right

- The best way to debug your matrices dimensions is by a pencil and paper.
- Dimension of `W` is `(n[l],n[l-1])` . Can be thought by right to left.
- Dimension of `b` is `(n[l],1)`
- `dw` has the same shape as `W`, while `db` is the same shape as `b`
- Dimension of `Z[l]`, `A[l]`, `dZ[l]`, and `dA[l]` is `(n[l],m)`

Why deep representations?

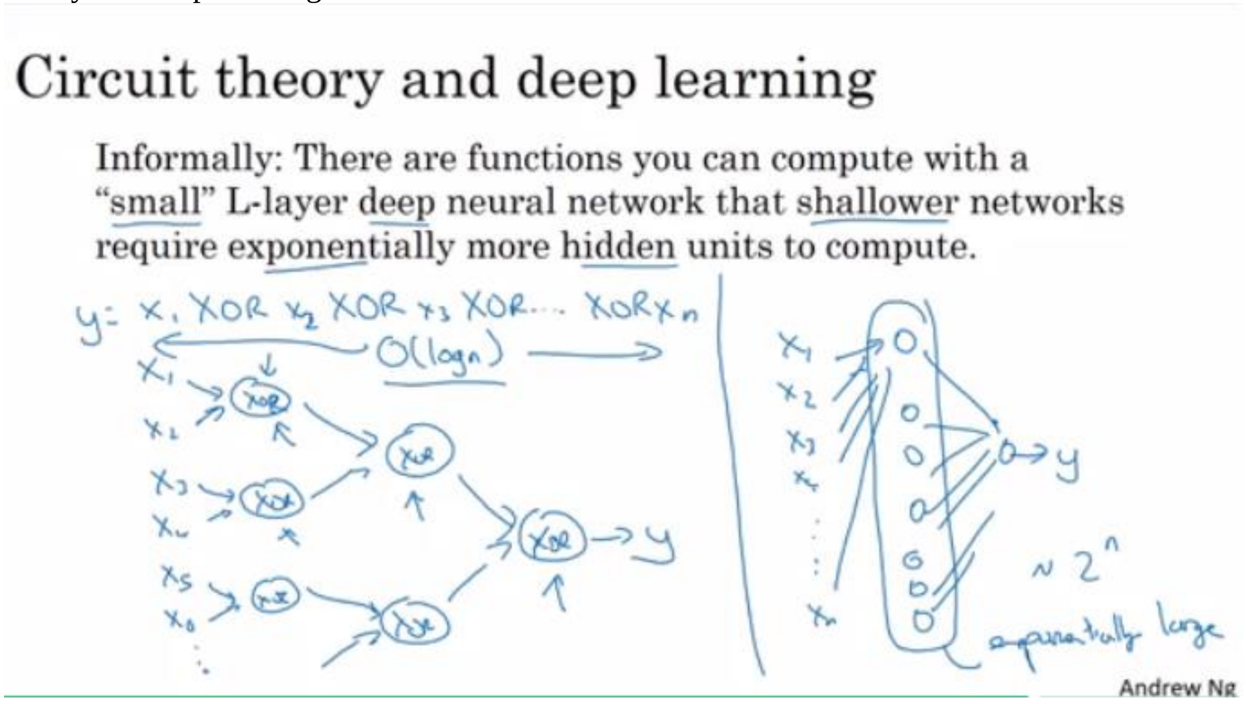
- Why deep NN works well, we will discuss this question in this section.
- Deep NN makes relations with data from simpler to complex. In each layer it tries to make a relation with the previous layer. E.g.:
  -
    1. Face recognition application:
       - Image ==> Edges ==> Face parts ==> Faces ==> desired face

- ◦

    2. Audio recognition application:

        ▪ Audio ==> Low level sound features like (sss,bb) ==> Phonemes ==> Words ==> Sentences

- Neural Researchers think that deep neural networks "think" like brains (simple ==> complex)
- Circuit theory and deep learning:



Circuit theory and deep learning

Informally: There are functions you can compute with a "small" L-layer deep neural network that shallower networks require exponentially more hidden units to compute.

    ◦

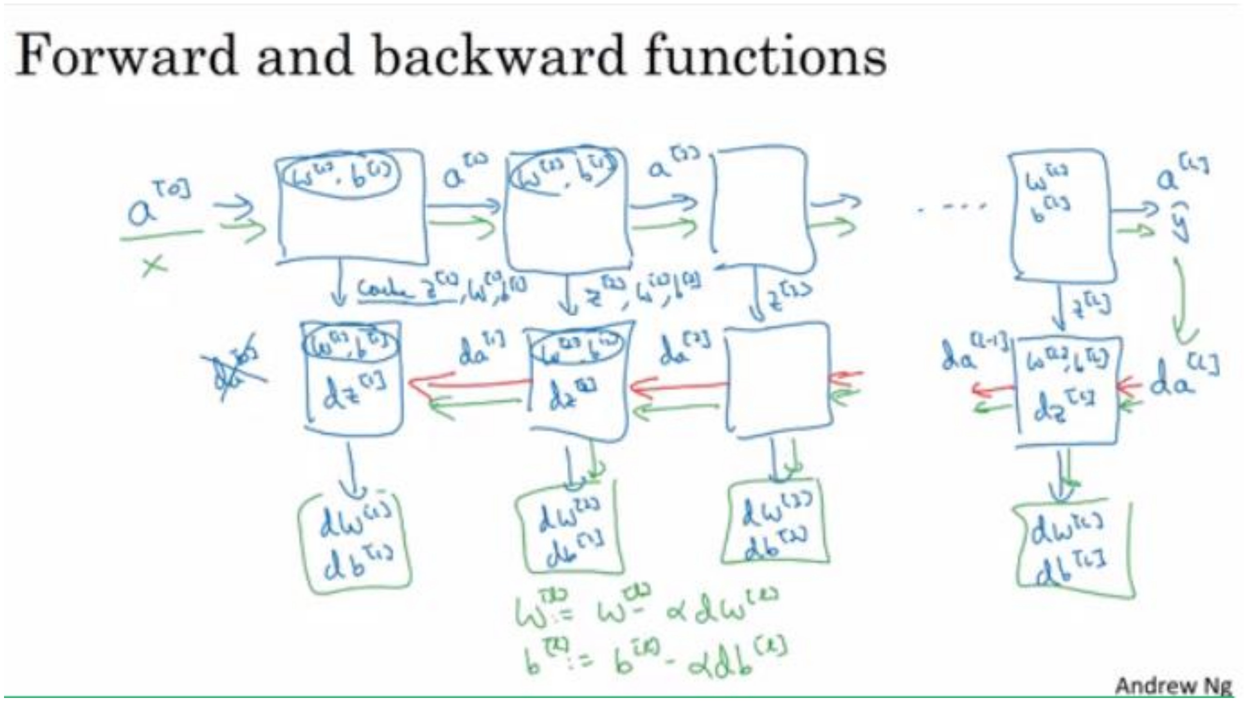
- When starting on an application don't start directly by dozens of hidden layers. Try the simplest solutions (e.g. Logistic Regression), then try the shallow neural network and so on.

Building blocks of deep neural networks

- Forward and back propagation for a layer l:

```
                  ----------
                 |  Forward |
    A[l-1] ==>   |   Using: | ==>A[l]                Equations: Z[l] = W[l]A[l-1] + b[l]   then A[l] = g[l](Z[l])
                 |    W[l]  |
                 |    b[l]  |
                  ----------
                      |
                      |  cache Z[l]
                      |  cache A[l-1]
                  ----------
                 | Backward |
dA[l-1]<==       |   Using: |   <== dA[l]              Equations: dZ[l-1] = (W[l].T * dZ[l]) * g'[l](Z[l-1])
                 | W[l],b[l]|
                 |    Z[l]  |
                  ----------
                     |
                     | dW[l]                          Equations: dW[l] =  (dZ[l] * A[l-1].T) / m
                     | db[l]                          Equations: db[l] =  sum(dZ[l]) / m  # Sum over rows
```

    ◦

- Deep NN blocks:
  - 
    Forward and backward functions
$a^{[0]}$
x
$w^{[l]}, b^{[l]}$
$a^{[l]}$
cache $z^{[l]}, w^{[l]}, b^{[l]}$
$\hat{y}$
$da^{[l]}$
$dz^{[l]}$
$dw^{[l]}$
$db^{[l]}$
$w^{[l]} := w^{[l]} - \alpha dw^{[l]}$
$b^{[l]} := b^{[l]} - \alpha db^{[l]}$
Andrew Ng

Forward and Backward Propagation

- Pseudo code for forward propagation for layer l:
- `Input  A[l-1]`
- `Z[l] = W[l]A[l-1] + b[l]`
- `A[l] = g[l](Z[l])`
- `Output A[l], cache(Z[l])`
- Pseudo code for back propagation for layer l:
- `Input da[l], Caches`
- `dZ[l] = dA[l] * g'[l](Z[l])`
- `dW[l] = (dZ[l]A[l-1].T) / m`
- `db[l] = sum(dZ[l])/m                # Dont forget axis=1, keepdims=True`
- `dA[l-1] = w[l].T * dZ[l]            # The multiplication here are a dot product.`
- `Output dA[l-1], dW[l], db[l]`
- If we have used our loss function then:
- `dA[L] = (-(y/a) + ((1-y)/(1-a)))`

Parameters vs Hyperparameters

- Main parameters of the NN is `W` and `b`
- Hyper parameters (parameters that control the algorithm) are like:
  - Learning rate.
  - Number of iteration.
  - Number of hidden layers `L`.
  - Number of hidden units `n`.

    - Choice of activation functions.
- You have to try values yourself of hyper parameters.
- In the earlier days of DL and ML learning rate was often called a parameter, but it really is (and now everybody call it) a hyperparameter.
- On the next course we will see how to optimize hyperparameters.

What does this have to do with the brain

- The analogy that "It is like the brain" has become really an oversimplified explanation.
- There is a very simplistic analogy between a single logistic unit and a single neuron in the brain.
- No human today understand how a human brain neuron works.
- No human today know exactly how many neurons on the brain.
- Deep learning in Andrew's opinion is very good at learning very flexible, complex functions to learn X to Y mappings, to learn input-output mappings (supervised learning).
- The field of computer vision has taken a bit more inspiration from the human brains then other disciplines that also apply deep learning.
- NN is a small representation of how brain work. The most near model of human brain is in the computer vision (CNN)

Extra: Ian Goodfellow interview

- Ian is one of the world's most visible deep learning researchers.
- Ian is mainly working with generative models. He is the creator of GANs.
- We need to stabilize GANs. Stabilized GANs can become the best generative models.
- Ian wrote the first textbook on the modern version of deep learning with Yoshua Bengio and Aaron Courville.
- Ian worked with OpenAI.com and Google on ML and NN applications.
- Ian tells all who wants to get into AI to get a Ph.D. or post your code on Github and the companies will find you.
- Ian thinks that we need to start anticipating security problems with ML now and make sure that these algorithms are secure from the start instead of trying to patch it in retroactively years later.

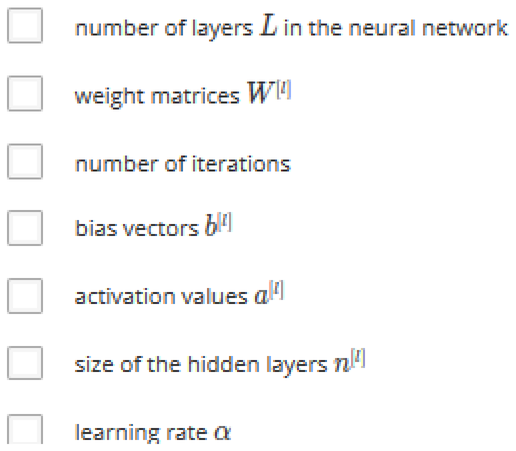
Week 4 Quiz - Key concepts on Deep Neural Networks

1. What is the "cache" used for in our implementation of forward propagation and backward propagation?
    - ☐ It is used to cache the intermediate values of the cost function during training.
    - ☑ We use it to pass variables computed during forward propagation to the corresponding backward propagation step. It contains useful values for backward propagation to compute derivatives.

- ☐ It is used to keep track of the hyperparameters that we are searching over, to speed up computation.
- ☐ We use it to pass variables computed during backward propagation to the corresponding forward propagation step. It contains useful values for forward propagation to compute activations.

  the "cache" records values from the forward propagation units and sends it to the backward propagation units because it is needed to compute the chain rule derivatives.

2. Among the following, which ones are "hyperparameters"?



☐ number of layers L in the neural network

☐ weight matrices $W^{[l]}$

☐ number of iterations

☐ bias vectors $b^{[l]}$

☐ activation values $a^{[l]}$

☐ size of the hidden layers $n^{[l]}$

☐ learning rate α

   (Check all that apply.) **I only list correct options.**

   - size of the hidden layers n[l]
   - learning rate α
   - number of iterations
   - number of layers L in the neural network

   Note: You can check this Quora post or this blog post.

3. Which of the following statements is true?
   - [✓ The deeper layers of a neural network are typically computing more complex features of the input than the earlier layers.] Correct
   - ☐ The earlier layers of a neural network are typically computing more complex features of the input than the deeper layers.

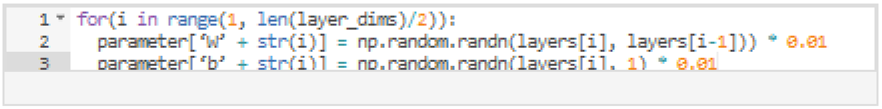
   Note: You can check the lecture videos. I think Andrew used a CNN example to explain this.

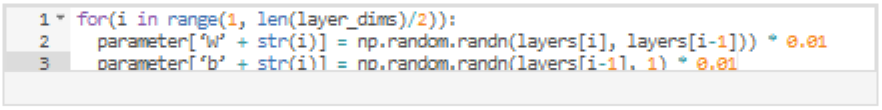
4. Vectorization allows you to compute forward propagation in an L-layer neural network without an explicit for-loop (or any other explicit iterative loop) over the layers l=1, 2, …,L. True/False?

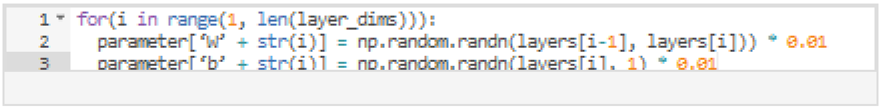
   - True
   - [✓ False]

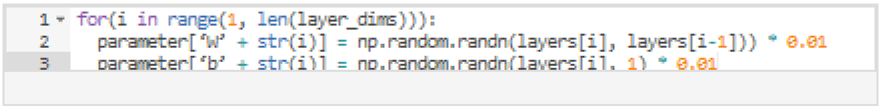
   Note: We cannot avoid the for-loop iteration over the computations among layers.

5. Assume we store the values for $n^{[l]}$ in an array called layers, as follows: layer_dims = [n_x, 4,3,2,1]. So layer 1 has four hidden units, layer 2 has 3 hidden units and so on. Which of the following for-loops will allow you to initialize the parameters for the model?

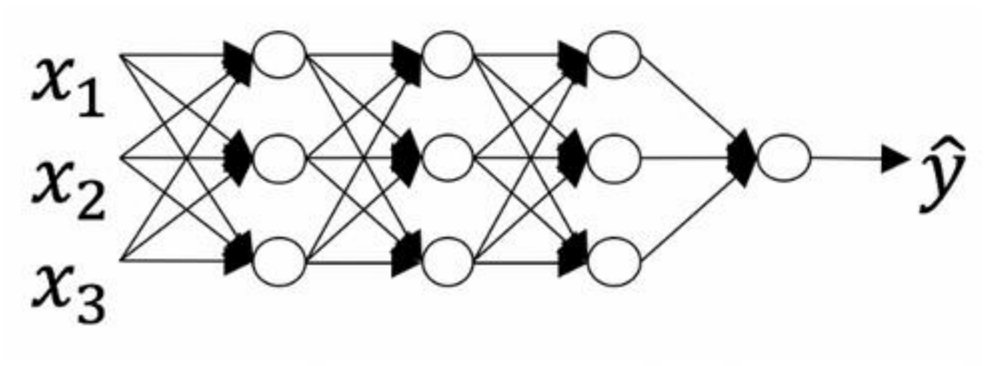
○ 
```
for(i in range(1, len(layer_dims)/2)):
  parameter[‘W’ + str(i)] = np.random.randn(layers[i], layers[i-1])) * 0.01
  parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) * 0.01
```

○ 
```
for(i in range(1, len(layer_dims)/2)):
  parameter[‘W’ + str(i)] = np.random.randn(layers[i], layers[i-1])) * 0.01
  parameter[‘b’ + str(i)] = np.random.randn(layers[i-1], 1) * 0.01
```

○ 
```
for(i in range(1, len(layer_dims))):
  parameter[‘W’ + str(i)] = np.random.randn(layers[i-1], layers[i])) * 0.01
  parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) * 0.01
```

● 
```
for(i in range(1, len(layer_dims))):
  parameter[‘W’ + str(i)] = np.random.randn(layers[i], layers[i-1])) * 0.01
  parameter[‘b’ + str(i)] = np.random.randn(layers[i], 1) * 0.01
```

6. Consider the following neural network.



How many layers does this network have?

- [✓ The number of layers L is 4. The number of hidden layers is 3.]
- The number of layers L is 3. The number of hidden layers is 3.
- The number of layers L is 4. The number of hidden layers is 4.
- The number of layers L is 5. The number of hidden layers is 4.

7. During forward propagation, in the forward function for a layer l you need to know what is the activation function in a layer (Sigmoid, tanh, ReLU, etc.). During backpropagation, the corresponding backward function also needs to know what is the activation function for layer l, since the gradient depends on it. True/False?

- [✓ True]
- False

During backpropagation you need to know which activation was used in the forward propagation to be able to compute the correct derivative.

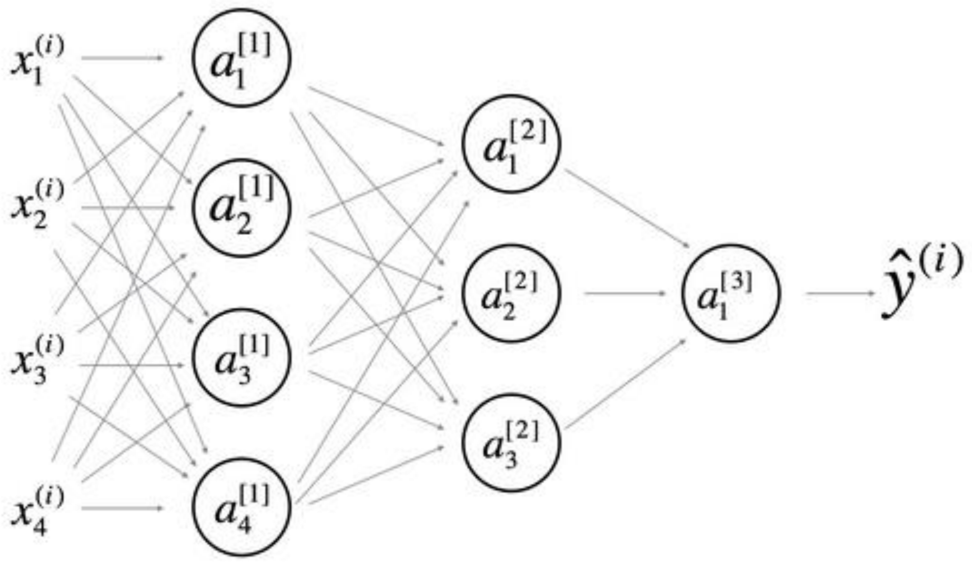
8. There are certain functions with the following properties:

(i) To compute the function using a shallow network circuit, you will need a large network (where we measure size by the number of logic gates in the network), but (ii) To compute it using a deep network circuit, you need only an exponentially smaller network. True/False?

- [✓ True]
- False

Note: See lectures, exactly same idea was explained.

9. Consider the following 2 hidden layer neural network:



Which of the following statements are True? (Check all that apply).

- ☐ $W^{[1]}$ will have shape (4, 4)
- ☐ $b^{[1]}$ will have shape (4, 1)
- ☐ $W^{[1]}$ will have shape (3, 4)
- ☐ $b^{[1]}$ will have shape (3, 1)
- ☐ $W^{[2]}$ will have shape (3, 4)
- ☐ $b^{[2]}$ will have shape (1, 1)
- ☐ $W^{[2]}$ will have shape (3, 1)
- ☐ $b^{[2]}$ will have shape (3, 1)
- ☐ $W^{[3]}$ will have shape (3, 1)
- ☐ $b^{[3]}$ will have shape (1, 1)
- ☐ $W^{[3]}$ will have shape (1, 3)
- ☐ $b^{[3]}$ will have shape (3, 1)

- W^[1] will have shape (4, 4)
- b^[1] will have shape (4, 1)
- W^[2] will have shape (3, 4)
- b^[2] will have shape (3, 1)
- b^[3] will have shape (1, 1)
- W^[3] will have shape (1, 3)

Note: See this image for general formulas.

10.

Whereas the previous question used a specific network, in the general case what is the dimension of W^[l], the weight matrix associated with layer l?

- W^[l] has shape (n^[l],n^[l−1])

Note: See this image for general formulas.