Congratulations! You passed!

Grade received 100%

Latest Submission Grade 100% To pass 80% or higher

Go to next item

1. Which of the following are true? (Check all that apply.)

1/1 point

 $a^{[2](12)}$ denotes the activation vector of the 2^{nd} layer for the 12^{th} training example. ${\color{red} {\checkmark}}$ Correct $a^{[2](12)}$ denotes activation vector of the 12^{th} layer on the 2^{nd} training example. ${\color{red} {}}$ X is a matrix in which each row is one training example. $a^{[2]}$ is the activation output of the 2^{nd} layer for the 4^{th} training example. ${\color{red} {\checkmark}}$ X is a matrix in which each column is one training example.

 $igspace{ \ \ \ } a_4^{[2]}$ is the activation output by the 4^{th} neuron of the 2^{nd} layer

✓ Correct

✓ Correct



Correct
 Great, you got all the right answers.

2. The tanh activation is not always better than sigmoid activation function for hidden units because the mean of its output is closer to zero, and so it centers the data, making learning complex for the next layer. True/False?

1/1 point

False

O True

∠⁷ Expand

⊘ Correct

Yes. As seen in lecture the output of the tanh is between -1 and 1, it thus centers the data which makes the learning simpler for the next layer.

3. Which of the following is a correct vectorized implementation of forward propagation for layer 2?

1/1 point

$$\bigcirc \ \ Z^{[2]} = W^{[2]} \, X + b^{[2]} \\ A^{[2]} = g^{[2]} (Z^{[2]})$$

$$igcirc Z^{[1]} = W^{[1]} \, X + b^{[1]} \ A^{[1]} = g^{[1]} (Z^{[1]})$$

$$egin{aligned} Z^{[2]} &= W^{[2]}\,A^{[1]} + b^{[2]} \ A^{[2]} &= g(Z^{[2]}) \end{aligned}$$

Expand

⊘ Correct

Yes. The elements of layer two are represented using a superscript in brackets.

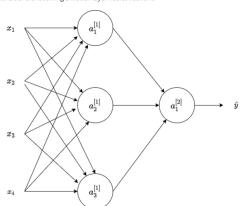
The States Consider the Solicular becomes the default activation has no derivative at c = 0 this rarely causes any problems in practice. Moreover it has become the default activation function in many cases, as explained in the lectures. Consider the Solicular default activation function in many cases, as explained in the lectures. Consider the Solicular default activation function in many cases, as explained in the lectures. Consider the Solicular default activation function in many cases, as explained in the lectures. Consider the Solicular default activation function in many cases, as explained in the lectures. Consider the Solicular default activation function in many cases, as explained in the lectures. Consider the Solicular default activation function function for the lectures. Play parameters are solicular default activation function function function for the hidder layer. Consider the Solicular default activation for each order own in the first timension is kept, thus (1, 2). Suppose you have built a necesification for each order own in the first timension is kept, thus (1, 2). Suppose you have built a necesification for each order own in the first timension. Tourifulate incomposition of the solicular development of the solicular default activation function for the hidden layer's nearers will be form different computations. The first hidden layer's nearers will deal different computation. The New Solicular the weights are most lively different, each nearers will do a different computation. The New Solicular the weights are most lively different, each nearers will do a different computation. The New Solicular to the indicate the weights of the computation of the computation. The Solicular to the computation of the computation of the lidden units. You initialize the weights or realized the weights and only one plant the computation of the		of the KeLU activation function is becoming more rare because the KeLU function has no derivative for True/False?	1/1 point
© Fable Consider the following code: **Topard** Consider the following code: **Topard** Consider the following code: **Topard** **Topard** Consider the following code: **Topard** **Topard** **Topard** **Topard** Consider the following code: **Topard** **			
Consider the following code: All paint Consider the following code: All paint All pa			
© Cervest Note Although the Bell Uninction has no derivative at c = 0 this rarely causes any problems in practice. Moreover it has become the default activation function in many cases, as explained in the loctures. 1/1 point 1/2 point 1/3 point 1/4 point 1/	(1)	False	
© Cervest Note Although the Bell Uninction has no derivative at c = 0 this rarely causes any problems in practice. Moreover it has become the default activation function in many cases, as explained in the loctures. 1/1 point 1/2 point 1/3 point 1/4 point 1/			
Consider the following code: # Region_sr python # Paparondom.cand(3, 2) # Paparondom.cand(3,			
The second of the set of function has no derivative at c = 0 this arealy causes any problems in practice. Moreover it has become the default activation function in many cases, as explained in the lectures. Consider the following code: #*begin_src python **a np.randon.rand(0, 2) **y np.sun(x, asten), keepdims-True) #*ernd_urc What will be yshape? 23	Z7	Expand	
ve. Athough the RetU function has no derivative at $c = 0$ this rarely causes any problems in practice. Moreover it has become the default activation function in many cases, as explained in the lectures. Consider the following code: **Degin_src python **Inequal_precipitation **Inequal_precipitation **Operation_precipitation **Operation_precipitation_precipitations from each other even in the first iteration. True/Palse? **Operation_precipitation_precipitations from each other even in the first iteration. True/Palse? **Operation_precipitation_precipitations from each other even in the first iteration. True/Palse? **Operation_precipitation_precipitations from each other even in the first iteration. True/Palse? **Operation_precipitation_precipitation_precipitations from each other even in the first iteration. True/Palse? **Operation_precipitation_preci			
Consider the following code: #begin_arc python * inp.random.rand(3, 2) y = np.sum(k, axi=0, keepdims=True) #end_sc What will be yshape? Ci Ci Ci Ci Ci Ci Ci C	_		
## begin_sic python x = np.random.rand(3, 2) y = np.sum(x, asit=0, keepdims=True) ## end_sic What will be yshape? (b) (c) (d) (d) (d) (d) (d) (d) (d	N	foreover it has become the default activation function in many cases, as explained in the lectures.	
## begin_sic python x = np.random.rand(3, 2) y = np.sum(x, asit=0, keepdims=True) ## end_sic What will be yshape? (b) (c) (d) (d) (d) (d) (d) (d) (d			
x = pp.random.rand(3, 2) y = pp.sum(x, axis=0, keepdims=True) #*end_sc What will be yshape? ② (3) ③ (1, 2) ③ (4, 3) ③ (7, 2) ③ (8) ⑤ (1, 2) ② (9) ⑥ (1, 2) ⑥ (1, 2) ⑥ (1, 2) ⑥ (1, 2) ⑥ (1, 2) ⑥ (2) ⑥ (3, 3) ② (2) ⑥ (3, 4) ② (2) ⑥ (1, 2) ⑥ (3, 5) ② (2) ⑥ (4, 2) ⑥ (4, 2) ⑥ (5, 2) ⑥ (6, 3) ③ (7, 2) ⑥ (7, 2) ⑥ (8, 1) ② (2) ⑥ (1, 2) ⑥ (1, 2) ⑥ (1, 2) ⑥ (1, 2) ⑥ (2) ⑥ (3, 3) ⑥ (4, 2) ⑥ (4, 2) ⑥ (5, 2) ⑥ (6, 3) ⑥ (7, 2) ⑥ (8, 1) Ø (8, 2) Ø (9, 2) Ø (1, 3) Ø (1, 4)	Consid	er the following code:	1/1 point
y = mp.sumfx, axis=0, keepdrims=True) ##end_src What will be yshape? (2)	#+begi	n_src python	
### Present Strom Present Stro	x = np.i	random.rand(3, 2)	
What will be yishape? ② (2) ③ (3) ③ (1, 2) ③ (3, 1) ② Correct ③ Correct Yes. By choosing the axis=0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims="True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each neuron will do a different computation. ⑤ To be Yes. Since the weights are most likely different, each neuron will do a different computation. ⑥ To be Yes. Since the weights are most likely different, each neuron will do a different computation. ② Correct ② Correct ② Correct ③ True ③ True ③ True ③ True ③ True ⑤ True ③ True ③ True ③ True ③ True ③ True ③ True ⑤ So long as you initialize the weights randomly gradient descent is not affected by whether the weights are largor or small. ⑤ The weights are largor or small. ⑤ The large values, using pp_andom.rand(r_,)*1000. What will happen? ⑤ So long as you initialize the weights randomly gradient descent is not affected by whether the weights are largor or small. ⑤ The weights are largors of the tent h to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	y = np.:	sum(x, axis=0, keepdims=True)	
 ② (2) ② (1,2) ③ (1,2) ⑤ (1,2) ⑥ (1) Suppose (in a substitution of the sum is computed over each column of the array, thus the resulting array is a row-vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1,2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. To have. Since the weights are most likely different, each neuron will do a different computation. To pand Cerrect Cerrect So long as you infanctions in the hidden layers of a multilayer neural network is equivalent to using a single layer. Palse True Cerrect Cerrect Cerrect Cerrect Cerrect You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np, andom, and nc,1*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or semall. The will cause the length of the tenth to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	#+end_	src	
 (a) (b) (c) (c) (d) (e) (e) (f) (e) (f) (g) (g)	What w	rill be y.shape?	
 (a) (b) (c) (c) (d) (e) (e) (f) (e) (e) (f) (e) (e) (f) (e) (f) (f) (g) (g)		(2)	
 ⑤ (1, 2) ⑥ (3, 1) ☑ (3, 1) ☑ Correct (Yes. By choosing the axis=0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? ☐ False No. Since the weights are most likely different, each neuron will do a different computation. ☑ True Yes. Since the weights are most likely different, each neuron will do a different computation. ☑ True Yes. Since the weights are most likely different, each neuron will do a different computation. ☑ True Yes. Since the weights are most likely different, each neuron will do a different computation. ☑ True Yes. Since the weights are most likely different, each neuron will do a different computation. ☑ True Yes. Since the weights are most likely different, each neuron will do a different computation. ☑ True Yes. Since the weights are large or since the indicate of the tent in also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. ☑ This will cause the injunts of the tent in also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. 	_		
Correct Yes. By choosing the axis=0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. 1/1 point layer. True/False? False True True/False? False True True/False? You have built a network using the tanh activation function for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(-,,,-)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the injusts of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.			
✓ Correct Yes. By choosing the axis=0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False?			
 ✓ Correct Yes, By choosing the axis=0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first heration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. Palse True Yes. Since the weights are most likely different, each neuron will do a different computation. You pand Correct You pand You be part of the tent the indident layers of a multilayer neural network is equivalent to using a single layer. You have built a network using the tanh activation function g(c) = c is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using numadom and for the layer or small.			
Yes. By choosing the axis-0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. ✓ Expand ✓ Correct Using linear activation functions in the hidden layers of a multilayer neural network is equivalent to using a single layer. True/False? False True ↑ True 1/1 point 1/2 point 1/2 point 1/2 point 1/3 point 1/4 point 1/5 point 1/6 point 1/6 point 1/6 point 1/7 point 1/7 point 1/7 point 1/8 po	27	Expand	
Yes. By choosing the axis-0 the sum is computed over each column of the array, thus the resulting array is a row vector with 2 entries. Since the option keepdims=True is used the first dimension is kept, thus (1, 2). Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. Such decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. ✓ Correct Using linear activation functions in the hidden layers of a multilayer neural network is equivalent to using a single layer. True/False? False True True 1/1 point 1/2 point 1/2 point 1/3 point 1/4 po			
Suppose you have built a neural network with one hidden layer and tanh as activation function for the hidden layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. ✓ Correct Using linear activation functions in the hidden layers of a multilayer neural network is equivalent to using a single layer. True/False? False True True True True 1/1 point 1/2 point 1/2 point 1/1 point 1/2 point 1/3 point 1/4 point	_		
layer. You decide to initialize the weights to small random numbers and the biases to zero. The first hidden layer's neurons will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different computation. Correct Using linear activation functions in the hidden layers of a multilayer neural network is equivalent to using a single layer. True/False? False True			
Using linear activation functions in the hidden layers of a multilayer neural network is equivalent to using a single layer. True/False? False True True Expand Correct Yes. When the identity or linear activation function $g(c) = c$ is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	neuron	s will perform different computations from each other even in the first iteration. True/False? False No. Since the weights are most likely different, each neuron will do a different computation. True Yes. Since the weights are most likely different, each neuron will do a different	
Using linear activation functions in the hidden layers of a multilayer neural network is equivalent to using a single layer. True/False? False True True Expand Correct Yes. When the identity or linear activation function $g(c) = c$ is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(,)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	_		
layer. True/False? ☐ False ☐ True ☐	⊚ (VI 1544	
layer. True/False? ○ False ○ True Correct Yes. When the identity or linear activation function $g(c) = c$ is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(.,.,.)*1000. What will happen? ○ So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. ○ This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.			
 True ② Correct Yes. When the identity or linear activation function g(c) = c is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(,)*1000. What will happen? ③ So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. ④ This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. 	layer. T	rue/False?	1/1 point
 ✓ Correct Yes. When the identity or linear activation function g(c) = c is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(.,,)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. 			
 ✓ Correct Yes. When the identity or linear activation function g(c) = c is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(.,.,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. 		ilue	
 ✓ Correct Yes. When the identity or linear activation function g(c) = c is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(.,.,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. 			
 ✓ Correct Yes. When the identity or linear activation function g(c) = c is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(.,.,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow. 			
Yes. When the identity or linear activation function $g(c)=c$ is used the output of composition of layers is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	27	Expand	
is equivalent to the computations made by a single layer. You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	⊘ 0	orrect	
You have built a network using the tanh activation for all the hidden units. You initialize the weights to relatively large values, using np.random.randn(.,,,)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.			
large values, using np.random.randn(,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.	1:	e equilation to the computations muce by a single tayer.	
large values, using np.random.randn(,.)*1000. What will happen? So long as you initialize the weights randomly gradient descent is not affected by whether the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.			
the weights are large or small. This will cause the inputs of the tanh to also be very large, thus causing gradients to be close to zero. The optimization algorithm will thus become slow.			1 / 1 point
close to zero. The optimization algorithm will thus become slow.			
become large. You therefore have to set α to a very small value to prevent divergence; this	0	This will cause the inputs of the tanh to also be very large, thus causing gradients to also	



 $Yes.\ tanh\ becomes\ flat\ for\ large\ values;\ this\ leads\ its\ gradient\ to\ be\ close\ to\ zero.\ This\ slows\ down\ the$

 $\textbf{9.} \ \ \ \text{Consider the following 1 hidden layer neural network:}$

optimization algorithm.



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



 $b^{[1]}$ will have shape (3, 1).

✓ Correct

Yes, $b^{[k]}$ is a column vector and has the same number of rows as neurons in the k-th layer.

ightharpoonup *E: $b^{[2]}$ will have shape (1,1)

✓ Corre

Yes. $b^{[k]}$ is a column vector and has the same number of rows as neurons in the k-th layer,

 $igwedge W^{[1]}$ will have shape (3, 4).

✓ Corre

Yes. The number of rows in $W^{[k]}$ is the number of neurons in the k-th layer and the number of columns is the number of inputs of the layer.

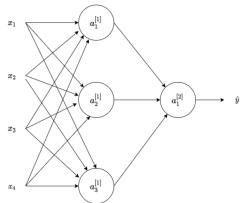
 $\ \ \ \ \ b^{[1]}$ will have shape (1, 3)

Z Expand

⊘ Correct

Great, you got all the right answers.

 $\textbf{10.} \ \mathsf{Consider} \ \mathsf{the} \ \mathsf{following} \ \mathsf{1} \ \mathsf{hidden} \ \mathsf{layer} \ \mathsf{neural} \ \mathsf{network} ;$



What are the dimensions of ${\cal Z}^{[1]}$ and ${\cal A}^{[1]}$?

 $\bigcirc \quad Z^{[1]} \text{ and } A^{[1]} \text{ are (4, m)}$

1/1 point

1 / 1 point

∠⁷ Expand

igodots correct Yes. The $Z^{[1]}$ and $A^{[1]}$ are calculated over a batch of training examples. The number of columns in $Z^{[1]}$ and $A^{[1]}$ is equal to the number of examples in the batch, m. And the number of rows in $Z^{[1]}$ and $A^{[1]}$ is equal to the number of neurons in the first layer.