Quiz-05

- Due Feb 16 at 11:59pm
- Points 10
- Questions 10
- Available Feb 14 at 6pm - Feb 16 at 11:59pm
- Time Limit None
- Allowed Attempts 3

Take the Quiz Again

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	120 minutes	6.5 out of 10
LATEST	Attempt 2	120 minutes	6.5 out of 10
	Attempt 1	1,691 minutes	3 out of 10

Correct answers are hidden.

Score for this attempt: 6.5 out of 10
Submitted Feb 16 at 3:53pm
This attempt took 120 minutes.

Question 1
1 / 1 pts

Consider the Following Hypothetical:

You've just finished reading Kaiming He's paper and are motivated to create a novel weight initialization method for Deep Learning. You settle upon using a **zero-mean uniform distribution for weight initialization** for weight initialization. Following the Forward Propagation scenario outlined in He's paper, which value for the 'b' parameter is suitable, given that **the uniform distribution ranges from** $[-b, +b]$ and 'n' **represents the number of neurons in each layer**?

Refer to the detailed analysis in the paper: https://arxiv.org/pdf/1502.01852.pdf (https://arxiv.org/pdf/1502.01852.pdf)

- sqrt(4/n)
- sqrt(2/n)
- sqrt(6/n)

Sufficient condition as per the Forward Propagation case is 0.5*n*Var(w) = 1. The derivation of the sufficient condition makes a number of independence assumptions and only requires the distribution to be 0 mean and symmetric. Zero-mean Uniform is a symmetric distribution. Therefore all we need to do is make sure the variance of the uniform distribution used to initialize w satisfies the above sufficient condition.

- sqrt(8/n)

Question 2
1 / 1 pts

In a neural network, a specific subset of the parameters $(w_1, w_2, w_3, \cdots, w_L)$ are all constrained to be identical in value, i.e. $w_1 = w_2 = w_3 \cdots = w_L = w^S$. Which of the following represents a valid SGD pseudocode to update their values when the network is being trained. Here T is the number of training instances in the batch, X_t is the t-th training instance and Loss(X_t) is loss computed for the t-th training instance. For each instance assume the forward and backward passes (to compute derivatives) have been performed before the "for i = 1:L" loop. Also assume $dLoss(X_t)/dw_i$ is computed using backdrop.

Hint: **(Lecture 9: Slides 65 - 74)**

- for t = 1:T
  $w^S = 0$
  for i = 1:L
  $w_i = w_i - \eta * dLoss(X_t)/dw_i$
  $w^S = w^S + w_i$
  for i = 1:L
  $w_i = w^S$

- for t = 1:T
  for i = 1:L
  $w^S = w^S - \eta * dLoss(X_t)/dw_i$
  for i = 1:L
  $w_i = w^S$

- for t = 1:T
  for i = 1: L
  $w_i = w_i - \eta * dLoss(X_t)/dw_i$

for t = 1:T

for i = 1:L

$w^S = w^S - \eta * dLoss(X_t)/dw_i$

$w_i = w^S$

Question 3
1 / 1 pts

You are given the problem of “wake-up word detection” -- the problem of recognizing if the wake-up word “IDLgod” has been spoken in a recording. You know “IDLgod” takes less than half a second to say, so you build a little “IDLgod” classification MLP that can analyze a half-second segment of speech and decide if the wake-up word has been spoken in it. You plan to scan incoming audio in chunks of half a second, sliding forward 25 milliseconds at a time, to detect if the wake-up word has occurred in any chunk.
Unfortunately, the training data you are given only consists of longish 15-20 second long segments of audio. The recordings tagged as “positive” data have the word “IDLgod” spoken somewhere in them, but the precise location is not marked. You are not even informed *how many* times the word is spoken in the recording -- all you know is that somewhere in that long recording are one or more instances of the wake-up word. So you have no way of slicing out the precise half-second segments where the word was spoken, to train your MLP, should you decide to do so.

The negative recordings do not have the word in them at all.

How would you train a model using this data? Keep in mind that during test time, you want to be able to classify incoming audio in chunks of exactly half a second.

Hint: **(Lecture 9: Slides 4 - 46)**

For each input of T seconds, construct a large network with (about) 40T copies of the MLP, with all of their outputs going into a softmax. Analyze the entire input with the large network, such that that the i-th copy of the MLP within the large network “looks” at the half second section of time starting at t = ((i-1)25+1) ms. Train the entire network over the complete recordings using KL divergence w.r.t. the labels for the inputs using regular gradient descent, where gradients of network parameters are computed using backprop.

Slice up the input into half-second segments, and treat all segments derived from the positive recordings as positive instances to train your MLP.

For each input of T seconds, construct a large network with (about) 40T copies of the MLP, with all of their outputs going into a softmax. Analyze the entire input with the large network, such that that the i-th copy of the MLP within the large network “looks” at the half second section of time starting at t = ((i-1)25+1) ms. Train the entire network over the complete recordings using KL divergence w.r.t. the labels for the inputs, using gradient descent, where all the copies of the MLP within the larger network are constrained to share parameters (i.e. have identical parameters)

Manually inspect the positive instances, and find the precise boundaries of the wake-up word (and extract 0.5sec segments of audio that contain them) to “clean up” your training data and train your model.

PartialQuestion 4
0.5 / 1 pts

Which of these pseudocodes perform the operations of a 2D CNN ?

Hint: lec 9 slides 208-213

Recall that by “CNN” that we mean it scans a 2D image of size UxV for patterns. Don’t worry about edge cases or shapes : assume that all these code are syntactically correct and represent different networks, and that all variables Y are preallocated to size sufficiently greater than UxV to ensure no exceptions happen in the code, and that these preallocated values of Y (other than the input to the net) are initialized to 0.

In the codes below, the index “l” represents the layer index. The networks have L layers. N represents the number of filters in a layer and K is the size of the filters.

Assume Y(0,*) is the input to the network, where “*” represents the necessary set of variables required to represent the entire input. The number of indices represented must be inferred from the code. In the notation below, in the products of w and Y, when both are multi-dimensional arrays (tensors), the notation represents a scalar valued tensor-inner-product, where corresponding components of the w and Y tensors are multiplied, and the set of pair-wise products is added to result in a scalar.

```
for l = 1:L
  for u = 1:U
    for v = 1:V
      Y(l,u,v) = activation(w(l,u,v)Y(l-1,u,v))
Y = softmax({Y(:,U,V)})
```

```
for l = 1:L
  for u = 1:U
    for v = 1:V
      for m = 1:N
        Y(l,m,u,v) = \
          activation(w(l,m,:,:,:)Y(l-1,:,u:u+K,v:v+K))
```

- [x] `Y = softmax({Y(L,:,:,:)})`

```
for u = 1:U
  for v = 1:V
    for l = 1:L
      for m = 1:N
        Y(l,u,v) = activation(w(l,m)Y(l-1,u:u+2,v:v+2))
```

- [ ] `Y = softmax({Y(L,:,:)})`

```
for l = 1:L
  for m = 1:N
    for u = 1:U
      for v = 1:V
        Y(l,m,u,v) = \
          activation(w(l,m,:,:,:)Y(l-1,:,u:u+K,v:v+K))
```

- [x] `Y = softmax({Y(L,:,:,:)})`

```
for u = 1:U
  for v = 1:V
    for l = 1:L
      for m = 1:N
        Y(l,m,u,v) = \
          activation(w(l,m,:,:,:)Y(l-1,:,u:u+K,v:v+K))
```

- [x] `Y = softmax({Y(L,:,:,:)})`

Recall that weights are shared, and as such, feature maps generated by one layer are used by the next. So we cannot loop through the layers after looping through the image dimensions for this to be distributed, which is the very nature of a CNN.

Question 5
1 / 1 pts
Mark all statements that are true.

Hint: Lecture 10, slides 16-38

- [x] The pooling layers in Lecun's CNN model the processing of the C-layers discovered by Hubel and Wiesel
- [x] Yann Lecun's CNN is a supervised analog of Fukushima's Neocognitron
- [x] The convolutional layers in Lecun's CNN model the processing of S-layers discovered by Hubel and Wiesel
- [x] The sequence of processing steps in Lecun's CNN is derived biological inspiration from Hubel and Wiesel's studies

IncorrectQuestion 6
0 / 1 pts

A time-delay neural network has three convolutional layers followed by a softmax unit. The convolutional layers have the following structure:

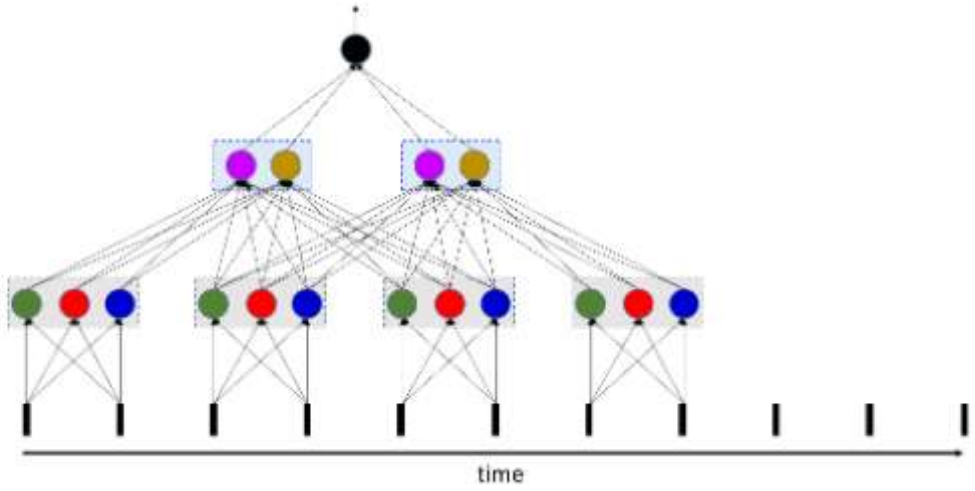
- The first hidden layer has 3 filters of kernel-width 2 and convolutions with a stride of 2;
- The second layer has 2 filters of kernel-width 3 and convolutions with a stride of 1;
- The third layer has 1 filter of kernel-width 2 and convolutions with a stride of 1.

As explained in class, the convolution layers of this TDNN are exactly equivalent to scanning the input with a shared-parameter MLP (and passing the set of outputs of the MLP at the individual time instants through a final softmax). What would be the architecture of this MLP?

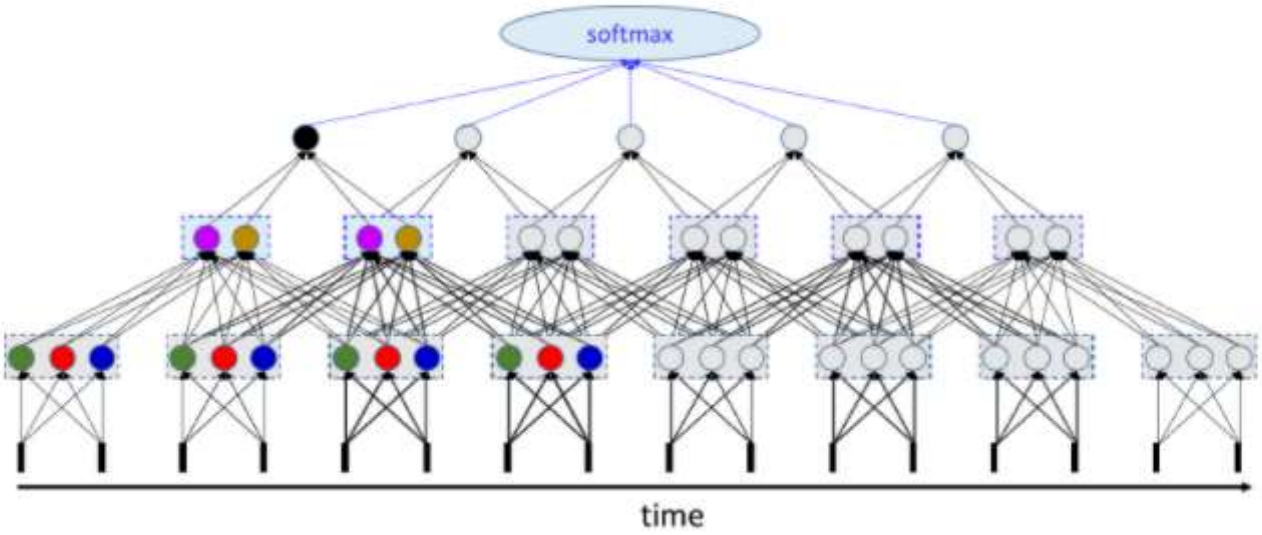
Hint: HW2P1 Section 8.1

- [ ] A three-layer MLP with 12 neurons in the first layer, 4 in the second, and 2 in the third
- [ ] Classification with this TDNN cannot be modeled as scanning with an MLP
- [ ] A three-layer MLP with 6 neurons in the first layer, 6 in the second, and 2 in the third
- [ ] A three-layer MLP with 12 neurons in the first layer, 4 in the second, and 1 in the third
- [x] A three-layer MLP with 3 neurons in the first layer, 2 in the second, and 1 in the third

The basic MLP is shown below. The sequence of little black bars shows the time sequence of input vectors. All blocks with the same background color are identical and share parameters (weights and biases). Neurons represented using the same color are identical and share parameters.



The full scan would have the structure shown below:



Incorrect Question 7
0 / 1 pts

A time-delay neural network has three convolutional layers followed by a softmax unit. The convolutional layers have the following structure:

- The first hidden layer has 4 filters of kernel-width 2;
- The second layer has 3 filters of kernel-width 2;
- The third layer has 2 filters of kernel-width 2.

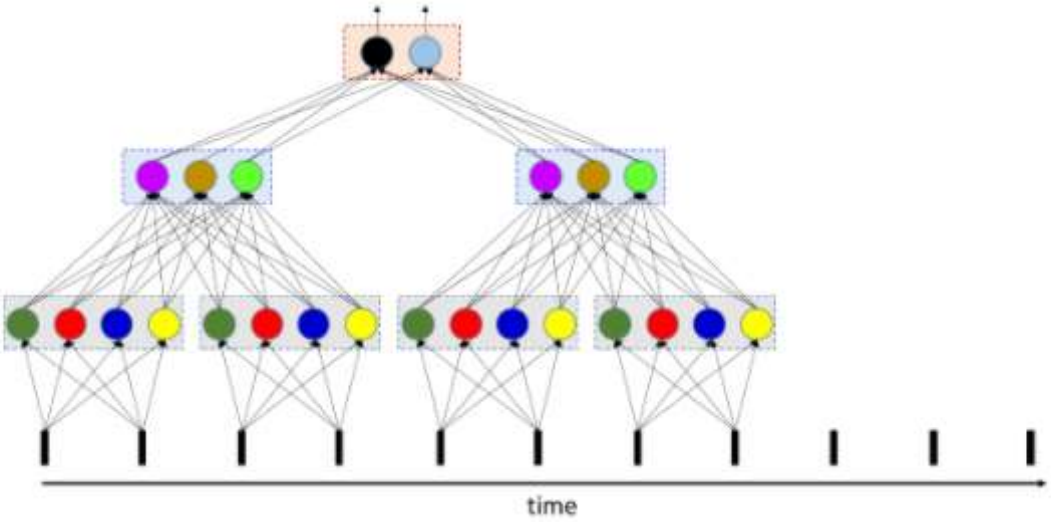
Assume that the stride of the convolution is 2 in every layer. As explained in class, the convolution layers of this TDNN are exactly equivalent to scanning the input with a (shared-parameter) MLP (and passing

the set of outputs of the MLP at the individual time instants through a final softmax). What would be the architecture of this scanning MLP?

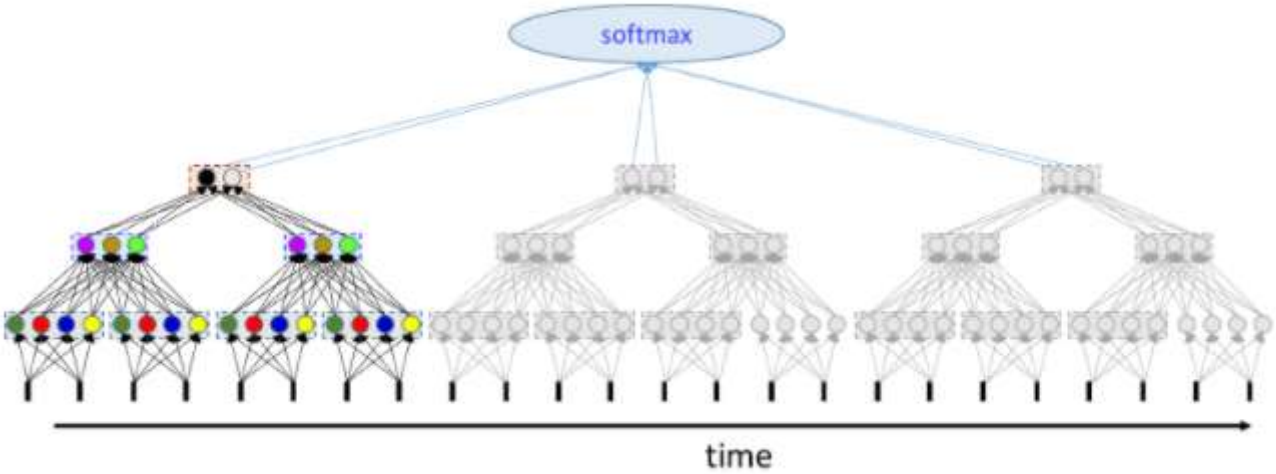
Hint: HW2P1 Section 8.1

- [ ] A one-layer MLP with a layer of size 24
- [x] A three layer MLP with 4 neurons in the first layer, 3 in the second layer and 2 in the third layer
- [ ] A three layer MLP with 8 neurons in the first layer, 6 neurons in the second layer and 4 neurons in the third layer
- [ ] Classification with this TDNN cannot be modeled as scanning with an MLP
- [ ] A three-layer MLP with 16 neurons in the first layer, 6 in the second layer and 2 in the third layer

The basic MLP is shown below. The sequence of little black bars shows the time sequence of input vectors. All blocks with the same background color are identical and share parameters (weights and biases). Neurons represented using the same color are identical and share parameters.



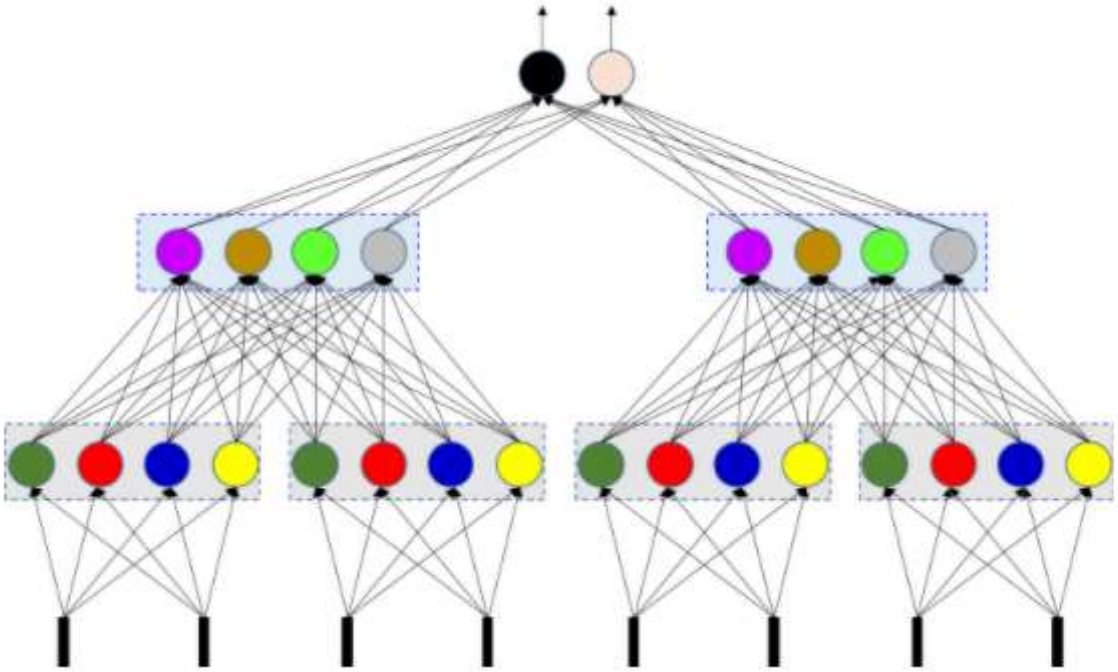
The full scan would have the structure shown below:



IncorrectQuestion 8

0 / 1 pts

A time-series input is scanned for a pattern using the following MLP with a stride of 4. Subsequently the outputs of the MLP at each time step are combined through a softmax. In the figure, neurons of identical color within each layer have identical responses.



The same operation can be performed using a time-delay neural network. What will the architecture of this TDNN be?

Hint: HW2P1 Section 8.1

- [ ] Three convolutional layers. 16 filters of kernel-width 16 in the first layer, with a stride of 16, 8 filters of kernel-width 8 with stride 8, and 2 filters of kernel-width 4 with stride 4 in layer 3.
- [ ] Three convolutional layers. 4 filters of kernel-width 2 in the first layer, with stride 2, four filters of kernel-width 2 in the second layer, with stride 2, and 2 filters of kernel-width 2 in the third layer, with a stride of 1.
- [x] Two convolutional layers and a flat layer. 4 filters of kernel-width 2 in the first layer, with stride 2. 4 filters of kernel-width 2 with stride 2 in the second layer. The final flat layer has two neurons with all the outputs of the second layer going to them.
- [ ] Three convolutional layers. 4 filters of kernel-width 2 in the first layer, with stride 2, four filters of kernel-width 2 in the second layer, with stride 2, and 2 filters of kernel-width 2 in the final layer. The information provided is insufficient to know the stride of the third layer.

Question 9

1 / 1 pts

Suppose a convolutional neural network for detecting flowers is trained on images that tend to have blue flowers in the upper left corner and pink flowers in the lower right corner. It is then tested on two different sets of test instances, testset 1 with blue flowers in the upper left corner, and pink flowers in the lower right corner, testset 2 with pink flowers in the upper left corner and blue flowers in the lower right corner. If the final layer of the network is fully connected then:

- (●) The error rate of the network will be higher on the testset 2 than on testset 1
- ( ) The expected value of the error rate will be worse than the empirical test error rate
- ( ) The error rate of the network will be about the same on both test sets
- ( ) The error rate of the network will be lower on testset 2 than on testset 1

Even though the CNN learns the existence of the blue and pink flowers independent of the position, the final fully connected layer is not position-invariant: the training data is biased to blue flowers at the top left and pink flowers in bottom right so its respective weights expect this sort of input. Thus, the network performs worse upon testset 2 and the error rate is higher.

Question 10

1 / 1 pts

The feature maps in lower layers (layers closer to the input) of a convolutional neural network compared to higher layers (layers closer to the output) of a convolutional neural network tend to:

(The options below refer to lower layers, and how they differ from higher layers)

- (●) Have more localized features (like edges), with smaller spatial extents of the image represented
- ( ) Have more global features (like shapes), with larger spatial extents of the image represented
- ( ) Have more localized features (like edges), with larger spatial extents of the image represented
- ( ) Have more global features (like shapes), with smaller spatial extents of the image represented

Quiz Score: 6.5 out of 10