ECE 498 ICC: IoT and Cognitive Computing

Spring 2020, Homework 1

SOLUTION

Question 1 (10 pts)

1. TensorFlow uses a dataflow graph (`tf.Graph`) to represent the computation. Explain the benefit of using dataflow graph to represent the computation in detail. [**2 pts**]

   - Parallelism: with dataflow graph, explicit data dependencies are presented clearly, which makes it easy for compiler to maximize parallelism in computation
   - Distributed execution: partitioning the computation to multiple devices becomes the problem of graph partitioning, which is a well-defined problem and has several existing solutions.
   - Compilation: the graph format enables compiler optimization e.g. fusion
   - Portability: The graph representation is language-independent. It can work as an intermediate representation (IR) for transformation optimization independent of the language.

2. List 2 advantages/disadvantages of using Cloud Computing vs Edge computing for an IoT system. [**4 pts**]

   - Cloud computing.

     Advantages: Can collect more data to perform analytics; Can perform tasks with high computation requirements.

     Disadvantages: High network bandwidth utilization; High latency; Must be connected to the network.
   - Edge computing.

     Advantages;

     Filter out unwanted data, reducing network usage; Low latency; Privacy of data.

     Disadvantages:

     Limited computation throughput; High energy consumption at end device.

3. For the following applications which processing (Cloud Computing or Edge computing) is more suitable? Briefly explain why. **[4 pts]**
   - Health data collected by an Apple watch/Fitbit to track your daily activities
   - Temperature sensors placed inside a refrigerated storage container to regulate the temperature during the shipping process
   - License plate readers at toll plazas
   - Medical wearable that detects when you fall.

- It depends on how students justify it.

Question 2: Tensorflow (10 pts)

1. $step(x)$ is defined as $step(x) = 0$ for $x < 0$ and $step(x) = 1$ for $x \geq 1$. Imagine that you are doing gradient descent with Tensorflow using a loss function $step(1-x)x^2 + step(x-1)[-(x-1)^2+1]$. x is a `tf.Variable` of type `tf.float32`. What happens:
   - when x is initialized to -2? **[3 pts]**
   - when x is initialized to $+3$? **[3 pts]**

   Assume that learning rate is very small. Justify your answer.
   - x converges to $x = 0$. Derivative for $x < 1$ is $2x$ which is negative for $x < 0$, and 0 at $x = 0$. Hence starting from $x = -2$, we keep subtracting a small negative number from x which moves it closer and closer to 0 at which point, the derivative is 0 and gradient descent stops.
   - x diverges to infinity. The derivative for $x >= 1$ is $-2x + 2$. For $x >= 3$, this is negative (non-zero). This means we keep adding a small positive number to x at each iteration, pushing it towards infinity. Hence gradient descend converges.

2. Assume that we are trying to solve an equation $x^2 + x = -1$ for scalar x, using gradient descent and TensorFlow. We setup x as `tf.Variable` of type `tf.float32`, and proceed to find the solution following the methodology in lab 1, part 3. Will this method be able to find the correct solution for this equation (Yes/No)? Prove or justify your answer. **[4 pts]**

   No, gradient descent will not find the correct solution. $x^2 + x + 1 = 0$ is a quadratic equation with no real-valued solutions (a quadratic equation has solutions $x = (-b \pm \sqrt{b^2 - 4ac})/2a$. Hence, gradient descend using a real valued variable, x (`tf.float32` is a real valued variable), cannot find the solution to the problem.

Question 3: Backpropagation (10 pts)

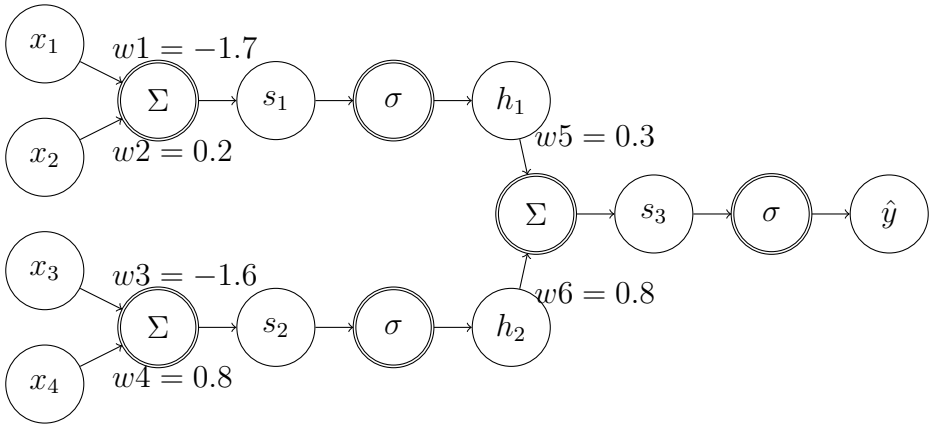

Figure 1: Neural network pipeline

Consider the neural network in Fig 1. Single-circled nodes denote variables (e.g. x_1 is an input variable, h_1 is an intermediate variable, $\hat{y}$ is an output variable), and double-circled nodes denote functions (e.g. Σ takes the sum of its inputs, and σ denotes the logistic function $\sigma(x) = \frac{1}{1+e^{-x}}$.
Suppose we have an MSE loss $L(y, \hat{y}) = \|y - \hat{y}\|_2^2$, We are given a data point $(x1, x2, x3, x4) = (0.3, 1.4, 0.9, -0.6)$ with true label 0.37. (Hint: : the gradient of an MSE loss function is $2\|y - \hat{y}\|$.)

1. Use the backpropagation algorithm to compute the partial derivative $\frac{\partial L}{\partial w_1}$, $\frac{\partial L}{\partial w_2}, \frac{\partial L}{\partial w_3}, \frac{\partial L}{\partial w_4}, \frac{\partial L}{\partial w_5}, \frac{\partial L}{\partial w_6}$. **[6 pts]**

   $s1 = -0.23$, $s2 = -1.92$, $h1 = 0.4428$, $h2 = 0.1279$, $s3 = 0.2352$, $\hat{y} = 0.5585$

   $\frac{\partial L}{\partial w_1} = 2 \times (0.5585 - 0.37) \times 0.2466 \times 0.3 \times 0.2467 \times 0.3 = 0.0021$

   $\frac{\partial L}{\partial w_2} = 2 \times (0.5585 - 0.37) \times 0.2466 \times 0.3 \times 0.2467 \times 1.4 = 0.0096$

   $\frac{\partial L}{\partial w_3} = 2 \times (0.5585 - 0.37) \times 0.2466 \times 0.8 \times 0.1115 \times 0.9 = 0.0075$

   $\frac{\partial L}{\partial w_4} = 2 \times (0.5585 - 0.37) \times 0.2466 \times 0.8 \times 0.1115 \times -0.6 = -0.0050$

   $\frac{\partial L}{\partial w_5} = 2 \times (0.5585 - 0.37) \times 0.2466 \times 0.4428 = 0.0412$

   $\frac{\partial L}{\partial w_6} = 2 \times (0.5585 - 0.37) \times 0.2466 \times 0.1279 = 0.0119$

2. Explain what are vanishing gradients and exploding gradients. You can watch this video before answering the question. (link)**[4 pts]**

   Open-ended question

Question 4: Neural Network (10 pts)

1. Consider two classification problems:
   Problem A: Items belonging to label '0' : {(0,0), (0,1)}, Items belonging to label '1': {(1,0), (1,1)}.
   Problem B: Items belonging to label '0' : {(0,0), (1,1)}, Items belonging to label '1': {(1,0), (0,1)}.
   Would linear classifiers be able to learn the patterns in these two classification problems? Justify your answer. **[2 pts]**

   Problem A can be solved by linear classifiers. Problem B cannot be solved by a linear classifier as it is not linearly separable.

2. For a classification problem with 16 feature inputs and 16 classes, compare the computational complexity of:
   (1) a deep network with eight hidden layers with 32 nodes each, and (2) a shallower network that has two hidden layers with 128 nodes each (all layers are fully-connected). What are the memory requirements of these two configurations? Assume ReLU activation is used in all layers except the output layer. Which one would you prefer to deploy on an edge device? **[2 pts]**

   For the 8 hidden layer network, we need $16 \times 32 + 32 \times 32 \times 7 + 32 \times 16 = 8192$ multiply-adds, which is a lot less than that for the 2 hidden layer network, which is $16 \times 128 + 128 \times 128 + 128 \times 16 = 20480$ multiply-adds. Similarly, the memory requirements for the weights and the feature maps can be calculated in the same way. The 8 hidden layer network requires much less memory space. The deeper network is preferred, because it is more light-weight. Training deeper networks can pose some challenges however that is not related to deployment.

3. You have trained a DNN model for an image classification application for static images, and you are deploying the model in the field. However, you find that the camera in the field was shifted in space compared to the camera used for capturing training images due to some logistic issues. What happens to the classification accuracy if you (1) used a Multilayer Perceptron as your model or (2) a Convolutional Neural Network as your model. Justify your answer. **[2 pts]**

   An MLP's accuracy will most likely drop even in very controlled situations (e.g., objects placed in a specific box during training etc.). A CNN is more tolerant towards accuracy drop, as convolutional kernels are linear shift invariant, and can extract features which are shifted geographically.

4. You are building a classifier and are comparing different activation functions. What are the trade-offs among the following three? (1) ReLU; (2) tanh; (3) unit step function? **[2 pts]**

Unit step cannot be differentiated, so it cannot be used for training. Tanh, ReLU may both be used functionally, but ReLU is faster computationally. There may be subtle slow convergence issues with ReLU because it has a non-differentiable point, but students don't need to point this out.

5. Are there any disadvantages in treating an N-class classification problem as a regression in N dimensions? Please explain. **[2 pts]**

   Cross-entropy loss cannot be used in a regression setting. Cross entropy loss helps faster convergence when there is a larger deviation from the input values as described in the lecture slides.

Question 5: Stassen's Algorithm (15 pts)

In machine learning applications, one of the most basic operations is matrix multiplication. Many so-called "AI chips" have large, dedicated on-chip regions for accelerating matrix multiplications. While Professor Hwu will teach us about parallel algorithms of matrix multiplications on GPU, there are fast algorithms that mathematicians had developed decades ago to out-perform naive $O(n^3)$ complexity. One of the earliest and most famous is Strassen's algorithm.
For matrices A, B, and $C \in R^{2^n \times 2^n}$, such that $C = AB$, we partition A, B, and C as follows:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix}$$

Strassen's algorithm defines new matrices M_k as follows:

$$\begin{aligned} M_1 &:= (A_{11} + A_{22})(B_{11} + B_{22}) \\ M_2 &:= (A_{21} + A_{22})B_{11} \\ M_3 &:= A_{11}(B_{12} - B_{22}) \\ M_4 &:= A_{22}(B_{21} - B_{11}) \\ M_5 &:= (A_{11} + A_{12})B_{22} \\ M_6 &:= (A_{21} - A_{11})(B_{11} + B_{12}) \\ M_7 &:= (A_{12} - A_{22})(B_{21} + B_{22}) \end{aligned}$$

and expresses C_{ij} in terms of M_k:

$$\begin{aligned} C_{11} &:= M_1 + M_4 - M_5 + M_7 \\ C_{12} &:= M_3 + M_5 \\ C_{21} &:= M_2 + M_4 \\ C_{22} &:= M_1 - M_2 + M_3 + M_6 \end{aligned}$$

1. Express the outcome matrix C_{ij} in terms of A_{ij} and B_{ij} as in the naive algorithm. **[2 pts]**

$$C_{11} := A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} := A_{11}B_{12} + A_{12}B_{22}$$
$$C_{21} := A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} := A_{21}B_{12} + A_{22}B_{22}$$

2. We now have A, B, and $C \in R^{4\times4}$; in other words, the matrices all have a dimension of 4×4. How many multiplications and additions are needed in the Strassen's algorithm? What about the naive algorithm? **[4 pts]**
Strassen: 49 multiplications and 198 additions, Naive: 64 multiplications and 48 additions

3. Based on what you have observed so far, compare Strassen's algorithm and the naive algorithm. **[3 pts]**
It takes less multiplications and thus reduces the big-O computation complexity. Multiplication is more expensive than addition, but Strassen's has more additions than naive algorithm.

4. Now we come back to the more basic case where A, B, and $C \in R^{2\times2}$. Let's try an alternative way to express the Strassen's algorithm. First, we reshape the input matrices A, B into vectors:

$$A = \begin{bmatrix} A_{11} \\ A_{12} \\ A_{21} \\ A_{22} \end{bmatrix}, B = \begin{bmatrix} B_{11} \\ B_{12} \\ B_{21} \\ B_{22} \end{bmatrix}$$

Then, we are able to express the outcome C as:

$$C = F \otimes ((G^T \otimes A) * (H^T \otimes B))$$

where $\otimes$ stands for matrix product, $*$ stands for element-wise product, and F, G, and $H \in R^{4\times7}$. We call F decoding matrix and G and H encoding matrices. Find F, G, H, and describe any observations on those matrices. **[6 pts]**
Hint: Look closely at the original equations. $M_1 = (A_{11} + A_{22})(B_{11} + B_{22})$ tells you that it is the *element-wise product* of $(A_{11} + A_{22})$ and $(B_{11} + B_{22})$. Also, you notice that only A_{11} and A_{22} appear in the left side of the product. Therefore, the first column of G is [1 0 0 1].

$$F = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & -1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & -1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Question 6: Curse of dimensionality and PCA

1. **Curse of dimensionality[2 pts]**
   Briefly explain the curse of dimensionality.

   - When number of dimensions increases the data is relatively sparse comparing to the volume of the feature space, therefore the model learning is also exponentially difficult. (Other reasonable explanations are also acceptable)

2. **Statistical interpretation of PCA[2 pts]**
   In lecture 4, you have learned the statistical interpretation of PCA. Briefly describe it below.

   - To find a limited set of directions (principal components) in the feature space, such that the variance of the input data along these directions are maximized. (Other reasonable explanations are also acceptable)

3. **Linear algebra interpretation of PCA[4 pts]**
   PCA can also be interpreted as: finding a subspace, represented with a set of orthonormal basis $v_1, ..., v_d$, such that the Frobenius norm of the difference between the original input and the projection onto the subspace is minimized.

   Given the description above, derive the objective function in terms of X, V, and Frobenius norm, where $V \in \mathbb{R}^{d \times k}$ whose column vectors are the orthonormal basis $v_1, ..., v_k$, and $X \in \mathbb{R}^{n \times d}$ whose rows are the input vectors $x_1, ..., x_n$. Notice that $V^T V = I$.

   Hint: What does matrix V do? How to derive the projection matrix onto a subspace given a set of basis? Does orthonormal basis make this projection it simpler?

   - $\text{argmin}_{V \in \mathbb{R}^{d \times K}} ||X^T - VV^T X^T||_F^2$

4. **Linear algebra derivation of PCA**

- **[6 pts]** Now that we have derived the loss function of PCA. Show that minimizing the the objective function can be written as maximizing $||XV||_F^2$
  You might need the following properties of Frobenius norm and trace:
  - $||A||_F^2 = tr(A^T A)$
  - $Ctr(A) = tr(CA)$ (C is a constant)
  - $tr(A + B) = tr(A) + tr(B)$
  - $tr(A^T) = tr(A)$
  - $tr(A^T A) = tr(AA^T)$

$$\begin{aligned}
||X^T - VV^T X^T||_F^2 &= \text{tr}((X^T - VV^T X^T)^T (X^T - VV^T X^T)) \\
&= \text{tr}(XX^T - 2XVV^T X^T + XVV^T VV^T X^T) \\
&= \text{tr}(XX^T) - 2\,\text{tr}(XVV^T X^T) + \text{tr}(XVV^T VV^T X^T) \\
&= ||X||_F^2 - 2\,\text{tr}(XVV^T X^T) + \text{tr}(XVV^T X^T) \\
&= ||X||_F^2 - \text{tr}(XVV^T X^T) \\
&= ||X||_F^2 - \text{tr}(V^T X^T XV) \\
&= ||X||_F^2 - ||XV||_F^2
\end{aligned}$$

Since we will take the argmin and $||X||_F^2$ is not related to V, we have $\text{argmin}\,||X^T - VV^T X^T||_F^2 = \text{argmax}\,||XV||_F^2$

- **[6 pts]** Now, we have $||XV||_F^2 = \text{tr}(V^T X^T XV)$. Assuming that $X^T X$ has an eigendecomposition $X^T X = QDQ^T$. We can then construct the vectors of V with the orthonormal eigenvectors of the matrix $X^T X$. That is, we can have $V = QZ$, where $Z \in \mathbb{R}^{k \times d}$ and $Z^T Z = I$.
  - First, state the reason why the eigendecomposition of $X^T X$ exists and the eigenvectors can form an orthonormal basis of the d-dimensional space.
  - Then, show that the objective function is can be transformed to $\text{tr}(Z^T DZ)$.

  Notice that, the maximum value of $\text{tr}(Z^T DZ)$ is the sum of the largest K eigenvalues of $X^T X$. (you don't need to prove this step)

  - $X^T X$ **is a symmetric matrix**, therefore it is guaranteed to have an eigendecomposition where the eigenvalues are real and eigenvecros can form an orthonormal basis of the whole n dimensional space.

–

$$\begin{aligned}\mathrm{tr}(V^T X^T X V) &= \mathrm{tr}((QZ)^T Q D Q^T (QZ)) \\ &= \mathrm{tr}((Z^T Q^T Q D Q^T Q Z) \\ &= \mathrm{tr}((Z^T I D I Z) \\ &= \mathrm{tr}((Z^T D Z)\end{aligned}$$

Question 7: IoT system design 2. (25 pts)

You are designing a security system. Your security system uses deep learning to recognize faces and detect unauthorized accesses, and a network of cameras, spread across 3 floors, each of size 3000 ft by 2000 ft. You are given that:

- Each camera sensor costs $10, has viewing radius of 100 ft, and generates 10 frames per second. Each frame is ∼3KB.
- You have a bandwidth of 10MB/s to the internet.
- You may choose from the following four devices. Their cost and the performance of the deep learning algorithm is given:

  Raspberry Pi: $55, Can run the deep network at 5FPS (frames per second)

  Xilinx PYNQ Z1: $200, 20FPS

  Nvidia Jetson TX1: $350, 35FPS

  Xilinx Cloud FPGA: $2000, 350FPS

- You may use only one type of communication protocol to communicate between devices. (Refer to class lectures for list of protocols and topologies)
- To detect an intrusion, you must look at 2 seconds of footage.

You must detect intrusions as soon as possible. How will you design such a system with minimal cost?

1. Provide a brief description of your system. You may provide a simple figure or block diagram. **[5 pts]**
2. How many cameras will you use? **[5 pts]**
3. What processing engine will you choose and how many cameras will be coupled to each? **[5 pts]**
4. What is your final cost? **[5 pts]**

5. What is the average worst-case latency (in milliseconds) for detecting an intrusion? [**5 pts**]

NOTE: You are allowed to make any reasonable assumptions and approximation about your solution. The points are not necessarily to get the exact solution, but to see how you would approach a problem at that scale using some back-of-the-envelope computation.

You can go about this in a few ways, but the goal is to be as fast as possible and minimize worst-case latency with minimal cost. To do this, you want the least number of cameras to cover the area, and minimal processing costs.

Cameras:

Camera viewing area is $3.14 * 100 * 100 = 31,400 ft^2$. So, to cover one floor, you need $(3000 * 2000)/(31400) = 191.0828 \sim= 192$ camera/floor. And a total of 576 cameras across 3 floors.

NOTE: This puts each camera at a distance of 200 ft from each other. Each camera generates: 10FPS, which needs a bandwidth of 30KB/s

Same examples:

Raspberry Pi: Not enough processing power for more than 1 camera. So we can directly couple one Pi with one Camera. For range of 200 ft You can use RFID via P2P network topology. Number of frames to be processed per camera = $2 * 10$

Processing Latency per frame = 1/5 = 200ms

Total latency = 20/5 = 4s + 200ms (For the 1st output) = 4200ms

Total Worst Case Latency = 4200ms

Cost = 576*(10 + 55) = $37,440

Perf/$ = 0.11ms/$

Pynq: Enough power for 2 camera, but both cameras will need to be linked to one PYNQ. We will assume that one Pynq couples with one camera, and another camera will send its video stream to the pynq. At 200m distance, we can use RFID and P2P. Worst case latency will be the cost of sending data to the Pynq and processing it.

Required bandwidth = 30KB/s (one camera is sending)

RFID bandwidth = 512 KB/s

Comm Latency per frame = 5.8ms (Assuming the camera will stream over images to the Pynq, we only need to wait on the 1st image, since we are not bottlenecked by comm bandwidth or processing throughput)

Processing latency per frame = 1/20 = 50ms

Processing latency = (2s * 20)/20 = 2s (2 cameras will deliver 20FPS of data, saturating the pipeline)

Total Worst case latency = Processing latency + Latency of 1st result + Latency of transfer

Total Worst Case Latency = 2000+50+5.8 = 2055.8ms

Total cost = 576*10 + (576/2 * 200) = $63,360

Perf/$ = 0.32ms/$

Cloud FPGA: Here we can assume all the cameras will send the streams to the cloud over WiFi. So we need to process at least 5760 FPS, so we need 17 FPGAs.
Total bandwidth needed = 30KB/s*576 = 17,280 KB/s = 17.2 MB/s
But we have only 10MB/s, which can stream 3333FPS, so we can only actually use 10FPGAs
Total time to transfer all frames = (2s * 10FPS * 576 * 3KB)/(1000KB/s) = 3.456s
Processing Latency = (576 * 2s * 10) /(10*350) = 3.29s
Total latency is MAX(cost to upload frames, processing time) since comm time dominates
Total worst case latency = 3456ms
Total cost = (576*10) + (10*2000) = $25760
Perf/$ = 0.134ms/$

TX1 This can be used in many ways. Since it has 35FPS, we can couple 3 or 4 cameras to one board. But we need to use a communication protocol with high bandwidth and that allows mesh/star networks. So we will explore 3 solutions:

- As we did with Pynq, we will couple 2 cameras to one TX1, with RFID via P2P:
  Processing latency per frame = 1/35 = 28.6ms
  Processing latency = (2s*20FPS)/35 = 1.1429s
  Comm latency per frame = 5.8ms
  Total latency = 1142.9 + 5.8 + 28.6 = 1177.3ms
  Total cost = $(576 * 10) + (350 * 576/2)$ = $106,560
  Perf/$ = 0.011ms/$

- 4 cameras per TX1. Use a star topology with the TX1 as the gateway. We will use Zigbee and star topology. For Bluetooth, we need to keep the cameras closer, and thus use more cameras.
  With Zigbee radius of 100m, we need cameras to operate on 50m radius, so we need (3000*2000)/(3.14*50*50) = 764.3312 = 765 cameras
  As, with cloud FPGA, we are limited by comm. Bandwidth.
  Total bandwidth needed = 30*4 = 120KB/s
  Zigbee bandwidth = 31.25KB/s
  Total time to transfer all frames = (2s * 10FPS * 4 * 3KB)/(31.5KB/s) = 7.62s
  Processing Latency = (4 * 2s * 10) /(35) = 2.29s
  Total latency is MAX(cost to upload frames, processing time)
  Total worst case latency = 7620s
  Total cost = (765*10) + (350*765/4) = $74,587.5 Perf/$ = 0.1021ms/$

- Or we can try to use Bluetooth, which has higher bandwidth, but less range. 4cameras in star topology with TX1 as gateway. For Bluetooth, we need to keep

the cameras closer, and thus use more cameras.
With Bluetooth radius of 30m, we need cameras to operate on 30m radius, so we need (3000*2000)/(3.14*30*30) = 2123.1423 = 2124 cameras
Total bandwidth needed = 30*4 = 120KB/s
Bluetooth bandwidth = 125KB/s
Total time to transfer all frames = $(2s * 10FPS * 4 * 3KB)/(125KB/s)$ = 1.92s
Time to transfer 1 frame from all cameras = (4*3KB)/(125KB/s) = 96ms
Processing Latency = (4 * 2s * 10) /(35) = 2.29s
Processing latency per frame = 1/35 = 28.6ms
Total latency = 2290 + 96 + 28.6 = 2414.6ms
Total cost = (2124*10) + (350*2124/4) = $207090
Perf/$ = 0.011ms/$