Cooperative Learning for Multi-Perspective Image Classification

Nick Nordlund
Electrical Engineering
Yale University
New Haven, USA
nicholas.nordlund@yale.edu

Heesung Kwon
US Army Research Laboratory
Adelphi, USA

Leandros Tassiulas
Electrical Engineering
Yale University
New Haven, USA

I. Abstract

Data gathered from dense sensor networks is often highly correlated across collocated sensors. For example, in video surveillance networks, multiple cameras can observe the same object from multiple angles. Despite the spatial and temporal dependencies between video frames from different cameras, the deep learning algorithms used in today's video analytics problems treat all frames as independent inputs to image classifiers and object detectors. The outputs of these classifiers and detectors on multiple frames are then fused to extract information about the underlying sensor region. We present a cooperative learning framework that allows sensors to train deep learning systems on their own local data and compressed insights from neighboring sensors' input data. This system fuses sensor data before classification to allow learning agents to more naturally handle correlated inputs and cooperate with neighboring sensors with minimal communication costs.

II. Introduction

In the era of Big Data, information is cheap but fusion is costly. As computational power moves closer to sources of data, sensor networks at the edge can become denser and more powerful. A sensor network is composed of a large number of sensor nodes and consists of sensing, data processing, and communicating components [1]. Sensors in these networks communicate data to a logically centralized controller that fuses data to give various insights into the observations. According to a 2017 marketing report from IBM, as much as 90% of all data in the world was created in the last two years alone [10]. With so much available data, environmental, medical, and military applications are becoming increasingly reliant on remotely sensed information. Not only are the number of sensors increasing, but the data each sensor produces is also growing more complex. For example, video surveillance networks generate thousands of frames of video data that need to be communicated to a central controller for processing. Transmitting video data may overwhelm the bandwidth capacity of the sensor network, and running object detectors and image classifiers at the controller may exceed its computational capacity. A growing amount of available data makes it more likely that useful information has been observed by the sensor network, but extracting those useful insights becomes increasingly more difficult.

In this paper, we use the video analytics problem as a paradigm example. Video analytics involves computationally expensive deep-learning algorithms that work in multiple stages. Object detectors first decompose video frames into multiple objects. Then, a series of image classifiers returns a more specific description of the objects in each video frame [11]. Prior works have addressed video analytics problems by leveraging processing power at the network edge. The method proposed in [13] involves locally running image classifiers and object detectors to process video frames when the offloading time exceeds the local processing time. In [8], the authors recognize that surveillance camera placement is usually very dense, so many cameras view the same regions of the sensing area. The authors utilize this redundancy and show that a Kalman filter can be applied to the results of multiple less-accurate lightweight image classifiers to identify frames that contain a query with great speed and accuracy. Prior works rely on using out-of-the-box image classifiers and object detectors.

These prior works illustrate that while the nature of the data is changing, the image classifiers' and object detectors' algorithms have largely remained the same. The deep learning algorithms only use batches of independent images for training. This process ignores the benefits of redundancy that a dense video surveillance network provides. Viewing the same object from multiple different angles intrinsically provides more information about it, and classifiers trained with more information generate more accurate results. Moreover, classifiers would produce more accurate results after fewer training epochs. Despite the advantages such training would provide, training is more computationally complex than the simple predictions that are currently a strain on the computational and communication capacities of today's sensor networks.

In this paper, we present a cooperative learning algorithm that combines multiple perspectives at the input to an image classifier and pushes the training stages to the network edge. Each camera in the network learns to communicate a summary of its video frames to nearby cameras, rather than broadcasting its entire video frame. These summaries are smaller than the original images and require less bandwidth to transmit over communication channels. This allows cameras to share a compressed version of their video data with all other cameras that observe the same target area. Each camera has access to the summaries from all other cameras but everything else is distributed. This differs from federated learning, a popular distributed learning scheme, in which there is a centralized model collectively updated by many agents. Instead, every node has its own model and nodes simultaneously learn to decode the summaries received from other nodes and encode their own summary of what they observe. In this paper, we show that a cooperative learning system that learns its own encoding and decoding scheme in a distributed manner is capable of achieving the accuracy of a fully centralized classifier with complete state information.

III. Background

A. Neural Nets for Image Classification

Deep learning relies on multi-layered neural networks that map inputs to desired outputs. The neural network can be thought of as a function $\hat{y} = f_\theta(x)$ where x is the input, $\hat{y}$ is the output, and $f_\theta(\cdot)$ is the network with parameters θ. Simply put, for image classification problems, the input to the neural network is the pixel values of an image and its output is an n-dimensional vector of values in the range $[0, 1]$. Each element in this output vector corresponds to a different class of object that the classifier is trained to recognize. Neural networks are trained with stochastic gradient descent according to Equation 1 where x represents a mini-batch of inputs and $L(\cdot)$ is a scalar loss function between the neural network output and the labeled data y.

$$\theta = \theta - \alpha \nabla_\theta L(f_\theta(x), y) \tag{1}$$

The stochastic gradient descent algorithm has been shown to consistently find a set of parameters that minimizes the overall loss function. Most research in computer vision now involves designing network architectures that best minimize the loss. The right sequences of convolutional and residual layers can result in significantly better accuracy [4] [6].

B. Federated Learning

Federated learning is a machine learning setting that trains a centralized model while keeping data distributed over a large number of clients [5]. Critically, data does not have to be identically distributed across the different clients for the system to converge to a minimal loss. The Federated Averaging algorithm is similar to the standard stochastic gradient descent. Each client trains a local copy of the central model on its own local data. After a number of update steps, the client shares its model parameters with the central controller. This controller averages the model parameters returned by all the clients and broadcasts the resulting model back to them. Federated learning demonstrates the ability of a distributed network of agents trained using a standard stochastic gradient descent with non-identically distributed data to cooperatively converge to an optimal solution.

IV. Problem

In a video surveillance setting, each camera in the network sees the same objects from different perspectives. Depending on the orientation and location of the target object, cameras running image classifiers on these different perspectives will receive outputs of varying degrees of confidence. Some cameras may have a clear view of the target and can be confident in their local image classifiers, while other cameras may have an awkward view of the target or have other objects in the scene obstructing their view. The following sections of this paper outline a solution to the following problem: If the cameras are allowed to communicate some information about their video frame at the current time step to all other cameras in the network, can the cameras reach a consensus on the objects they are observing, and importantly, can the local image classifiers achieve a more accurate result working cooperatively than they would achieve on their own?

A. Model Description

In our first set of experiments, we model a five camera system. We imagine one camera sees the target object head-on, while the other four see it from the top, bottom, left, and right respectively. We use the CIFAR-10 image set for training. The CIFAR-10 dataset is a dataset of 32×32 images of 10 classes of objects including cats, planes, cars, etc. The dataset contains 50,000 training images and 10,000 test images for validation. In the second set of experiments we use real video surveillance data from the PETS 2009 data set. This data set contains video data from multiple cameras observing the same intersection from different perspectives. The few thousand video frames show crowds of people of varying numbers moving across the intersection.

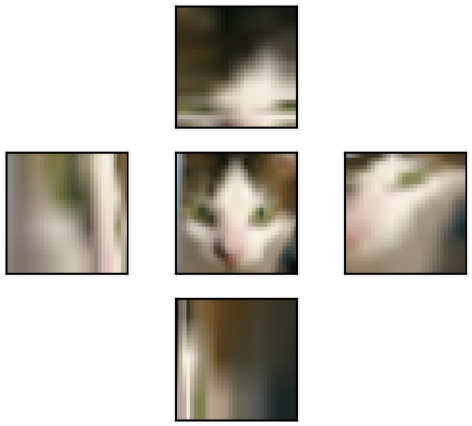

Fig. 1. Multiple perspectives of the same image. Original in center, surrounded by top, left, bottom right.

The CIFAR-10 dataset is not a multi-perspective image set. Instead we warp the images ourselves to simulate different perspectives. To accomplish this perspective shift, we first select four points from the original image at random. We ensure that the four points form a convex polygon and that the polygon is similar in shape to a trapezoid. We then distort the image such that the four corners of the original image map to the corners of the trapezoid. The orientation of the new perspective is represented by the longest side of the trapezoid. For example, if the long side of the trapezoid is on top, the perspective of the object is shifted to a view from the top. Lastly, we select the set of points contained within the largest square inscribed in the convex hull given by the four points and crop the image to this set. We resize this square to a full 32×32 image and use that as a new input. An example of the four new perspectives of a single image is shown in Figure 1.

Every perspective is warped randomly and unevenly, including the head-on view. Some perspectives might be so warped that they are difficult to classify even for a human. Each agent always receives a perspective from the same orientation. For example, one learning agent will always receive the top view. Each learning agent produces two outputs - a summary vector of information about the input that it sees and a classification based on the summary vectors it takes in from other agents in the sensing network. The learning agents are distributed and independent so they cannot perform backpropagation on the summary vectors communicated to them by the other agents.

While the PETS 2009 data set does contain multiple perspectives, only data from one camera view is labeled. Thus, we consider consecutive video frames as multiple perspectives of the same scene. This allows us to combine the current video frame with data from the past to increase accuracy in classification. Unlike the CIFAR-10 data that contains images that mostly depict a single object and have little background, the PETS 2009 dataset contains multiple objects in a large background. This tests a cooperative learning agent's ability to encode summary vectors that remain undiluted or uncontaminated by background clutter of multi-view videos.

B. Architecture

Each agent in the system has an identical architecture but are initialized with random parameters. Images in the CIFAR dataset are represented as 32×32 color images. We use a multi-layer convolutional neural network (CNN) for our image classifier. The full diagram of the network used is found in Figure 2. It is based on a simplified version of the VGGNet architecture [9]. The summary vector is the output of a fully-connected layer with 32 units. Video frames from the PETS 2009 set are an order of magnitude larger than images from the CIFAR-10 data set. For the PETS data, we use a simplified version of AlexNet. We keep the same overall structure of convolutional and fully-connected layers as AlexNet and add an additional fully-connected output layer for the summary vector of 64 units similar to Figure 2 [7]. The summary vector output for CIFAR-10 represents a 32-fold compression of the original image, and the vector for PETS 2009 is nearly 800-fold compression. We then concatenate this summary vector layer with summary vectors received from

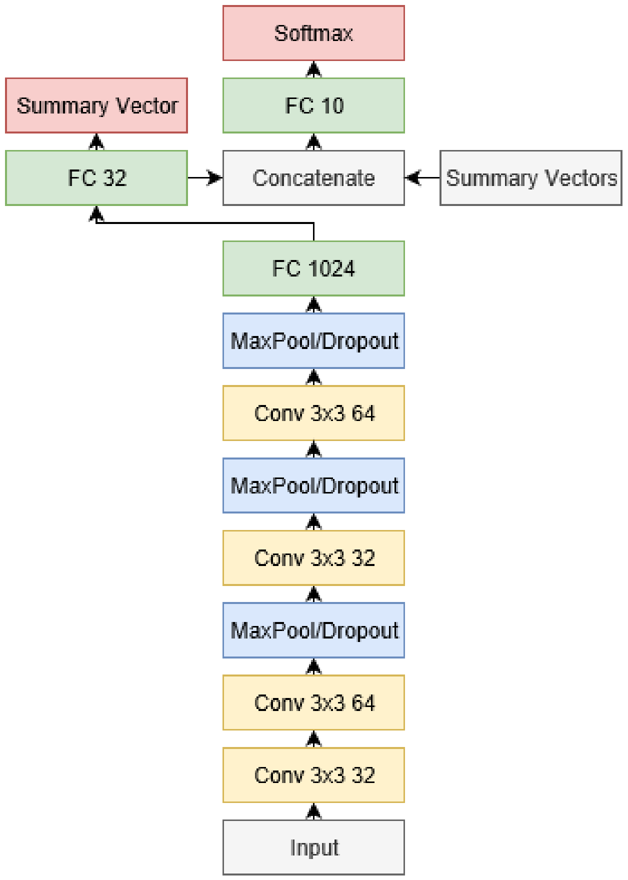

Fig. 2. Neural Network architecture for a single agent

other agents. The result of this concatenate layer is fed-forward through a final output layer of ten neurons corresponding to the ten possible object classes. ReLUs are used as activation functions for the hidden layers and the classification layer uses a softmax activation. A cross-entropy loss function and an Adam optimizer with a learning rate of .0001 are used.

C. Algorithm

The design of this problem allows us to use the simple stochastic gradient descent algorithm to find the network parameters for each learning agent. The stochastic gradient descent and backpropagation algorithms are used, but we prevent the gradients from propagating back through the summary vectors received from other agents [3]. This only allows each agent to optimize its own summary vector. Each agent can only react to changes in other agents' summary vectors.

V. Experiments

A. CIFAR-10

We compare the classification accuracy of four classifiers. All classifiers share the same basic neural network architecture. The first classifier is trained on an undistorted training set. This represents the highest accuracy our architecture can hope to achieve. The second classifier is trained on a distorted set of CIFAR-10 images without cooperation. This model represents

TABLE I
ACCURACY OF COOPERATIVE VS. NON-COOPERATIVE CLASSIFIERS ON CIFAR-10

Model	% Accuracy	% of Best-Case
Non-Cooperative Multi-Perspective	59.6	72.9
Non-Cooperative Single Perspective	81.7	100
Cooperative Multi-perspective	75.3	92.2
Cooperative Severe Perspective	71.7	87.8

an image classifier running on a central controller. All cameras offload their individual video frames to this central machine and it learns to classify them independently. A third classifier has all cameras work cooperatively on the five perspectives of a single image as shown in Figure 1. A fourth classifier learns without access to the head-on view, the easiest perspective to classify. If this classifier is successful, it demonstrates the ability of the cooperative system to learn by fusing unreliable image data rather than always listening to the camera with the most reliable perspective. A summary of the four models is as follows:

- **Non-Cooperative Multi-Perspective:** All frames from cameras are offloaded to a central node for training. The neural network treats all frames independently and does not consider the perspectives when training.
- **Non-Cooperative Single Perspective:** Another single classifier that trains on a central node. It takes undistorted images as its training data. This is used as a benchmark for the best performance our architecture can achieve.
- **Cooperative Multi-Perspective** Five separate agents train simultaneously on the five different perspectives - head-on, top, bottom, left, and right - and share summary vectors.
- **Cooperative Severe-Perspective** Four agents train simultaneously. The head on view is excluded to see how well agents are able to communicate and extract useful insights from highly distorted data.

Both cooperative learning models converge more quickly to a set of parameters that returns a more accurate classification than the model trained without cooperation. The five agent and four agent cooperative systems converge to 92% and 88% of the accuracy of the model trained the unaltered set of CIFAR-10 images. The model that does not fuse images from multiple perspectives and generates batches of images independently, however, only achieves 73% of the optimal accuracy in the same number of training steps. Each epoch represents a complete sweep over every image in the training dataset. A batch size of 128 image is used at every training step. The convergence rates of these learning systems is shown in Figure 3, and the final accuracy values are given in Table I.

B. PETS 2009

For this experiment, we train a classifier that counts the number of people in an image. Multiple perspectives will increase accuracy because there are objects in the background

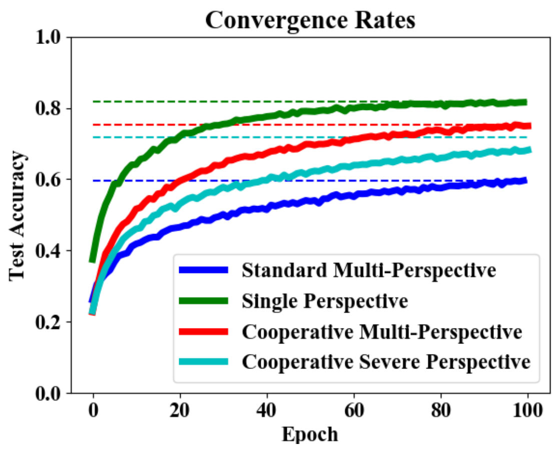

Fig. 3. Accuracy of cooperative learning vs. individual agents

that obstruct the camera's view. As mentioned in the previous section, we treat consecutive video frames as different perspectives. As a result, the classifier is able to infer the presence of occluded persons based on their trajectories observed in previous video frames. At every time step, the classifier takes in the current frame and previous four summary vectors as inputs and outputs a new summary vector and a classification. This architecture is a recurrent neural network trained without backpropagation through time (BTT). The current forward pass through the network cannot be used to train outputs from previous time steps. Instead the network must learn to decode its own outputs from the previous time steps. By eliminating the need for BTT, we reduce the complexity of the training step and allow these networks to train at the resource-limited network edge.

Crowd density estimation can be a challenging problem, but we use a simplified version to highlight the improvements our cooperative approach can have. We use a simplified AlexNet architecture for the basis of our classifier, as the authors of [12] chose. Their crowd-counting deep learning architecture partitions each image and learns to count the number of people in each partition. Their network outputs both a lower-resolution crowd-density map that shows where in the image more people are concentrated as well as a true count of the number of people. In our simplified approach, we remove the partitioning and density map elements of the problem and focus on just returning the true number of individuals present in a video frame. These simplifications allow image classifiers with a small number of parameters to achieve high levels of accuracy. Since there are not many objects that obstruct the camera's view in the PETS 2009 data set, we add random rotations, translations, and warps to augment the training data set. We use a batch size of 32 and an episode length of 300. We train a smaller modified AlexNet with 260,000 parameters instead of the actual 62,000,000 for 25 epochs [7]. The accuracy of this network trained cooperatively and on individual frames is shown in Table II.

The difference of five percentage points stays consistent

TABLE II
ACCURACY OF ALEXNET TRAINED WITH COOPERATIVE LEARNING VS. ALEXNET TRAINED WITH STANDARD ALGORITHM

Model	% Accuracy After 25 Epochs
Non-Cooperative	82.1
Cooperative	87.2

as we increase the size of the neural network. With more parameters comes higher accuracy, so a version using a million parameters, for example, achieved 95% accuracy cooperatively and 90% accuracy when training on single frame inputs. Since this is a simple problem, using a full-size AlexNet can achieve near perfect accuracy even in the non-cooperative case. This requires considerably more processing resources to reach the same accuracy that a smaller network trained cooperatively can achieve. Because the goal of a cooperative learning design is the push training closer to data, the neural networks we employ should be able to operate in resource-limited environments.

VI. CONCLUSION

In this paper, we present a cooperative supervised learning approach to video analytics problems. Unlike existing methods, our cooperative learning framework allows the agents to learn to encode spatial information about their environment. This not only eliminates the need to know the exact locations of all cameras in the sensor network and how their view angles overlap, but also opens up more possibilities for mobile surveillance networks, such as a network of body cameras on soldiers. While modern object detectors and image classifiers fuse their outputs to get meaningful information about the sensing region, the method we present here instead fuses the images before the final layers of the image classifier. We let the agents learn their own encoding/decoding strategies independently from other agents to reduce the communication costs of fusing video frames data before the output layer.

The methods outlined in this paper primarily utilize the spatial dependencies in a video surveillance network. Moving forward, we will seek to expand the cooperative learning algorithms to incorporate elements of [2]. This paper uses neural networks to approximate Kalman filter parameters online in order to account for non-linear motion of targets. This should further improve the accuracy of our cooperative object trackers. Lastly, we assume the network has enough communication capacity to handle all agents broadcasting their summary vectors to all other agents. Future work might have agents learn how relevant their partition is. Then, only agents with the most relevant partitions will communicate their summary vectors. This will reduce the communication costs.

One explanation of the success of cooperative learning algorithm set forth in this paper is that the agents are learning to treat the summary vector and their classifier output identically. Then, when each agent goes to make a classification decision, it simply chooses the outcome that the majority of its neighbors selected. This, however, may not be the case. In a separate experiment using the MNIST dataset, a dataset of 60,000 handwritten images, we partition each image into three disjoint sets. We then give three learning agents one of these sets and train an autoencoder using our cooperative learning methods. An autoencoder learns to compress the original image and recreate the original from the compressed vector. The results of the cooperative autoencoder for a 3×1 partition are depicted in Figure 4. Interestingly, the reconstructed images maintain most of the defining characteristics of the original image. The distributed autoencoders are not just learning to classify the images and reconstruct them based on the average image for the correct digit. The learning agents must therefore be learning a sophisticated way of encoding their summary vectors and decoding the information given to them by other agents in the network.

REFERENCES

[1] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. A survey on sensor networks. *IEEE Communications magazine*, 40(8):102–114, 2002.

[2] Huseyin Coskun, Felix Achilles, Robert DiPietro, Nassir Navab, and Federico Tombari. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5524–5532, 2017.

[3] Martin T Hagan and Mohammad B Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE transactions on Neural Networks*, 5(6):989–993, 1994.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[5] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[6] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[8] Nick Nordlund, Heesung Kwon, Geeth Ranmal De Mel, and Leandros Tassiulas. Image classification on the edge for fast multi-camera object tracking. In *2018 IEEE Military Communications Conference, MILCOM 2018, Los Angeles, CA, USA, October 29-31, 2018*, pages 1–5, 2018.

[9] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[10] D. Takaishi, H. Nishiyama, N. Kato, and R. Miura. Toward energy efficient big data gathering in densely distributed sensor networks. *IEEE Transactions on Emerging Topics in Computing*, 2(3):388–397, Sep. 2014.

[11] Victor Valls, Heesung Kwon, Tom LaPorta, Sebastian Stein, and Leandros Tassiulas. On the design of resource allocation algorithms for low-latency video analytics. pages 1–6, 10 2018.

[12] Cong Zhang, Hongsheng Li, Xiaogang Wang, and Xiaokang Yang. Cross-scene crowd counting via deep convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[13] Kevin Chan Zongqing Lu and Thomas La Porta. A computing platform for video crowdprocessing using deep learning. *Proceedings of IEEE International Conference on Computer Communications*, nov 2017.

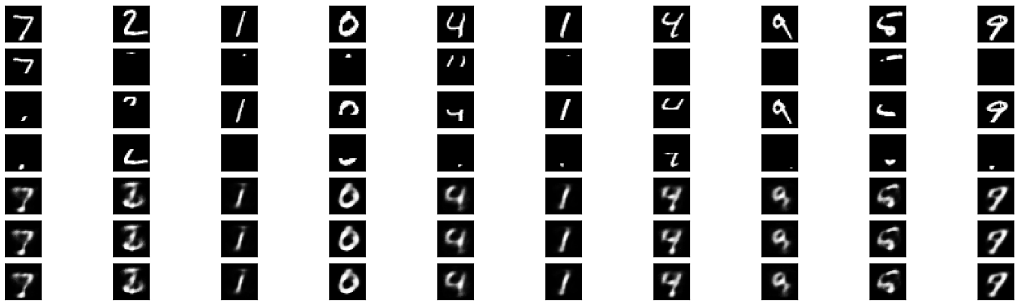

Fig. 4. Reconstructed images from partitions. Top row is original digit, next three rows are the partitions each agent gets as an input, and bottom three are the reconstructed images