Hierarchical Sparse Coding for Wireless Link Prediction in an Airborne Scenario

Stephen J. Tarsa and H.T. Kung
Harvard University, Cambridge, MA

***Abstract*—We build a data-driven hierarchical inference model to predict wireless link quality between a mobile unmanned aerial vehicle (UAV) and ground nodes. Clustering, sparse feature extraction, and non-linear pooling are combined to improve Support Vector Machine (SVM) classification when a limited training set does not comprehensively characterize data variations. Our approach first learns two layers of dictionaries by clustering packet reception data. These dictionaries are used to perform sparse feature extraction, which expresses link state vectors first in terms of a few prominent local patterns, or features, and then in terms of co-occurring features along the flight path. In order to tolerate artifacts like small positional shifts in field-collected data, we pool large magnitude features among overlapping shifted patches within windows. Together, these techniques transform raw link measurements into stable feature vectors that capture environmental effects driven by radio range limitations, antenna pattern variations, line-of-sight occlusions, etc. Link outage prediction is implemented by an SVM that assigns a common label to feature vectors immediately preceding gaps of successive packet losses; predictions are then fed to an adaptive link layer protocol that adjusts forward error correction rates, or queues packets during outages to prevent TCP timeout. In our harsh target environment, links are unstable and temporary outages common, so baseline TCP connections achieve only minimal throughput. However, connections under our predictive protocol temporarily hold packets that would otherwise be lost on unavailable links, and react quickly when the UAV link is restored, increasing overall channel utilization.**

I. Introduction

Wireless airborne networks face the challenge of providing fast reliable data delivery over flaky links. Link loss arises due to a confluence of factors, including signal strength attenuation, antenna pattern nulls, line of sight occlusions, and multipath interference. In general, these effects are determined by complicated environmental properties like geometry, and are exacerbated by node mobility. Accurate link state predictions in airborne scenarios like this would facilitate a host of network performance improvements, such as adaptative forward error correction to boost throughput and lower nodes' power consumption.

Predictive link models tend to be complex and require large amounts of input data. The two most common approaches are ray tracing and statistical modeling. Ray tracing uses detailed physical simulation to compute signal strength, and requires comprehensive geometric and propagation characteristics about the environment [1], [2]. In contrast, statistical modeling abstracts away physical effects by capturing structures such as correlations in training data. In this paper, due to the size and complexity of the flight environment, we pursue a statistical approach to predicting wireless air-to-ground links.

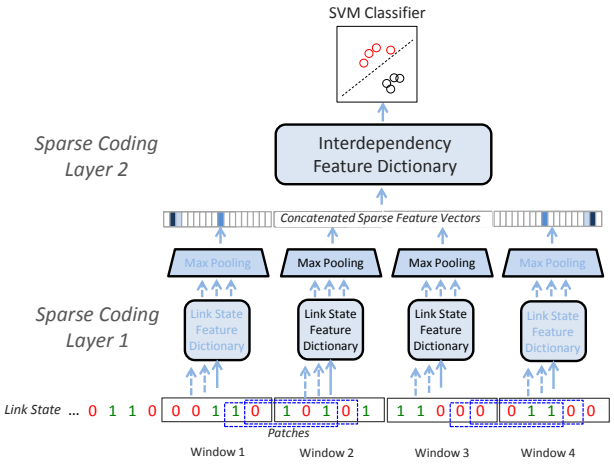

Fig. 1. Our two-layer hierarchical inference model extracts sparse feature vectors from five-packet-long windows in a first layer. Max pooling at Layer 1 selects the most prominent features from shifted overlapping patches within a window. A second layer extracts feature interdependency patterns from the concatenated pooled outputs of Layer 1. This model improves the accuracy of statistical tools like SVM classifiers when training data is modest relative to the degree of data variation.

The accuracy of a statistical model is limited by both the amount of available training data, and the correctness of modeling assumptions. Statistical link models in the literature include applications of Markov chains to capture the first and second order statistics for 802.11 and GSM networks [3], [4], [5], and latent variable Gaussian processes to map location-based signal strength statistics [6]. However, these methods are not well suited to capturing predictive correlations that span many packet times; the number of Markov chain state transitions that must be trained explodes with the length of link state sequences, and physical drivers of packet loss like occlusions violate Gaussian assumptions.

We will therefore use a data-driven hierarchical model based on the sparse coding framework, that combines clustering, sparse feature extraction, and non-linear pooling to exploit structure in link state information arising from environmental factors. Together, these techniques result in an approach that is tolerant to data variations. We show that our method improves the accuracy of statistical tools like Support Vector Machine (SVM) classifiers [7] when the amount of available training data is modest relative to the degree of data variation. As a result, we can perform inference tasks like outage prediction

based on effects lasting many packet times, without suffering onerous training data requirements or restrictive modeling assumptions. Furthermore, due to improved variation tolerance, we show that SVM classification accuracy holds across flights with different environmental conditions or node locations, without retraining.

Figure 1 illustrates our model, instantiated with a two layer configuration. Using sparse coding [8], the first layer extracts short patterns in packet losses, or features, from windows of link trace data. In order to improve tolerance to variations like small positional shifts, we combine the largest magnitude features from overlapping shifted patches within a window, a technique known as max pooling [9]. In this paper, these patches are sized equal to window size. The second layer then captures feature co-occurrence across consecutive regions of the flight path by sparse coding concatenated outputs from Layer 1. In this way, raw packet traces are transformed into a more-stable feature representation that characterizes the link in terms of a few canonical patterns. These feature vectors are fed to a top-level statistical inference module — in this case, an SVM that implements outage prediction by assigning a common label to vectors preceding gaps in packet delivery.

The model is initialized by training feature dictionaries for sparse feature extraction at Layers 1 and 2, in addition to the top level SVM. We cluster small patches of packet trace data using the K-SVD algorithm [10], and seed the Layer 1 dictionary with the learned cluster centroids. Training samples are then sparse coded at Layer 1 using Orthogonal Matching Pursuit (OMP) [11], and max-pooled. Concatenated outputs from Layer 1 feed a second round of clustering at Layer 2 to capture feature co-occurrence across neighboring windows.

Each technique used to transform link state data in our model improves variation tolerance for the top-level inference module. Clustering and sparse feature extraction together denoise link state measurements, and restore incomplete information [10]. Max pooling corrects for small variations like positional shifts that are difficult to control for during data acquisition. The model's overall hierarchical structure eases the difficulty of recognizing patterns over large spatial scales by independently characterizing local relationships, and then reusing the same training data to learn how those pieces fit together. Furthermore, in cases where the first layer feature dictionary is universal to all input data, only higher layers need be retrained to adapt the model for new environments. All of these general techniques also figure prominently in recent state-of-the-art computer vision results, such as face recognition under uncertain illumination [12], [13], [14], [15]. Interestingly, researchers have found evidence that similar techniques are employed in neuro biological systems like the visual cortex [16] [17]. To our knowledge, we are the first to apply this approach to predict wireless link quality.

Using this predictive model, we implement an adaptive link-layer protocol that improves aggregate channel throughput for TCP connections. Our protocol fixes intermittent packet drops with retransmissions, and queues packets to ride out predicted outages. This not only prevents TCP timeout when the UAV is out of range, but also avoids packet-loss-induced multiplicative decreases in TCP throughput at radio range edges, where links are available but unstable. As a result, connections react quickly when the UAV comes in range, and throughput is significantly increased.

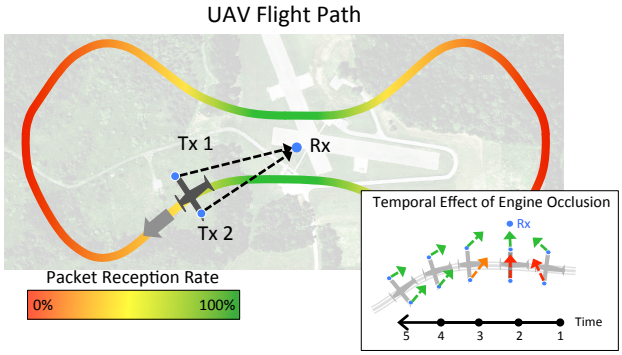

Fig. 2. A diagram of our UAV's flight path, with an illustration of typical packet loss rates. The UAV has two wing-mounted 802.11 transmitters that maintain independent TCP connections to a ground receiver. Inset, packet drops due to line-of-sight occlusion by the aircraft engine are depicted with red arrows. Occlusion is just one of several effects that cause temporary link outages.

The rest of this paper is organized as follows: in Section II, we describe our scenario and data measurement campaign. In Section III, we demonstrate the shortcomings of inference directly on packet traces, introduce hierarchical inference, and present our link model. In Section IV, we describe the adaptive link layer protocol, and in Section V, train the model and evaluate performance in emulation using trace replay.

II. UAV Scenario

Our UAV scenario is shown in Figure 2: a low-altitude fixed-wing aircraft flies multiple laps in a dumbbell shaped path over an airfield in upstate New York. Auto-pilot maintains positional consistency between laps, and GPS location variation is on the order of 15m when banking through turns. Two wing-mounted 802.11 b/g transmitters communicate with a ground node in the middle of the airfield. The property spans both fields and forest, and contains multiple vehicles and shelters. We target this scenario both because it is of tactical interest, and because results from the open interference-free environment can be corroborated by simple models, and serve as building blocks for more complicated scenarios.

We conducted two measurement campaigns, the first in June, and the second in October. In the June campaign, leaves and ground cover were in full bloom, and wind was minimal. In October, leaves had mostly fallen, temperatures were significantly lower, and wind was strong. The same flight path was used in both campaigns, however UAV speed fluctuated due to wind, the ground receiver's position was changed abitrarily, and signal propagation was subject to different environmental conditions [18]. During our experiments, transmitters broadcast 1500 byte packets at 1 Mbps, while receivers logged packets' SNRs and sequence numbers so

contiguous packet traces could be reconstructed.[1]

Characterizing the UAV's flight path using packet loss rates reveals three distinct regions: out-of-range, close-range, and intermediate. When out-of-range, packet loss is 100%, while at close range, it is lower than 10%. However, in the intermediate region, packet loss fluctuates wildly since transmissions are sensitive to occlusions from the aircraft engine, antenna nulls, multi path interference, etc. As a result, frequent gaps (i.e., link outages lasting multiple packet times) make it hard for off-the-shelf TCP implementations to maintain connections, and throughput often drops to zero outside the close-range regions. Our goal in this paper will be to improve data delivery in these intermediate regions by anticipating temporary outages.

The UAV link has inherent predictive power because physical factors cause correlations among packet transmissions, an effect known as "channel memory." For example, when the UAV engine blocks line-of-sight transmission, a pattern of consecutive packet losses may predict the success of a transmission in the next packet time. However, since UAV position and speed are not exactly repeatable, such an event is unlikely to have a precise unique signature observable in raw link data. Furthermore, while high level trends in link quality are stable between June and October data, packet reception rates change in magnitude, indicating that statistical methods sensitive to variation would require recalibration or additional training between flights.

III. A Predictive Link Model

A. Classification-Based Gap Prediction

To illustrate the sensitivity of off-the-shelf statistical inference tools to variations in raw link data, we first present a simple classification example. We train an SVM to label packet traces as belonging to one of two models: *noisy channel* or *temporary outage*. Input vectors are binary, with 0 indicating packet loss and 1 indicating reception. Consider an SVM trained on the following samples:

Noisy Channel Training Data:
(short intermittent strings of 0's)

[1111110111]
[0010101110]
[1101010011]

Temporary Outage Training Data:
(long strings of 0's)

[0000111111]
[1000011111]
[1111110000]

and the new channel observation:

[1111000011]

Intuitively, the new observation belongs to the temporary outage class, but an SVM labels it otherwise. This is because linear SVMs compute projections of training data to maximize separation between classes; if an unlabeled input vector is not a linear combination of training samples in its class, the projection gives unexpected results. In the UAV case, such a problematic variation could easily arise if the UAV's speed changes between training and evaluation flights. SVMs are a powerful tool, but this example illustrates the difficulty of variation tolerant inference when training data is limited.

B. Hierarchical Inference Background

To improve classification, and ultimately link prediction, we draw on advances that have appeared in several communities, including biologically-inspired computer vision [19], neural networks [20], and natural language processing [21]. Recent advances in these fields share several important insights:

1) *Hierarchical feature extraction* – By progressively transforming raw input data into feature vectors at larger and larger scales, complicated correlational relationships can be captured over large spatial ranges without training data requirements that are infeasible to meet.
2) *Extraction of sparse feature sets* – By latching signal patches to a small number of prominent exemplars, noisy or incomplete information is restored, and weakly expressed confounding information is discarded.
3) *Non-linear pooling of overlapping patches* – Similar to taking the maximum response of a convolutional filter, pooling improves tolerance to a broad class of minor variations like positional shifts that arise often in real-world data.

These general modeling techniques often appear alongside machine learning algorithms, which are used to find appropriate features in data that is difficult to model by hand.

Fascinatingly, neurobiology researchers have found evidence that these data processing techniques are active in the brain. The visual cortex is organized hierarchically into layers V1 through V4 that perceive visual cues at increasing levels of complexity. Meanwhile, neurons in these areas are known to fire sparsely in response to stimuli. And, the foveal system combines and attenuates the responses of bundled photoreceptor cells from different regions of the field of vision [22].

Applying these ideas to the example in Section III-A, we can show improved classification. We use a two level hierarchy similar to the one depicted in Figure 1 (accordingly analyzing only two windows of five packets for this input size). Layer 1 dictionaries are initialized to contain features corresponding to a four packet gap, no losses, and single losses at various positions. During training, Layer 2 dictionary features that group a four packet gap feature with a no-loss

[1]For a more detailed description of a similar campaign, see [4]. We graciously acknowledge those contributing researchers and engineers for their role in these experiments, including Chit-Kwan Lin, Dario Vlah, Dan Hague, Mike Muccio, Brendan Poland, Bob Gorman, and Jason Cassulis

feature will be labeled to the *temporary outage* class. Under this configuration, the new sample will be correctly labeled.

C. A Hierarchical Sparse Coding Channel Model

We now extend this methodology to real link data, yielding the full model shown in Figure 1. Our link data is again a contiguous 0/1 packet stream over a flight's duration, and is evaluated using a sliding window of 24 packet times. At Layer 1, 16 patches of 5 packets are enumerated, sparse coded, and pooled into 4 neighboring windows. Feature vectors from these windows are concatenated before being passed to Layer 2. Our top-level SVM assigns labels to training samples based on whether or not they precede a gap of 5+ packet times.

We use the sparse coding framework for dictionary learning and feature extraction [8], which solves the minimization:

$$\min \|X - DZ\|_2 \quad \text{s.t. } \|z_i\|_0 < k \text{ for } i = 1...n \qquad (1)$$

where X is an $m \times n$ matrix of n training samples (e.g., patches of size $m = 5$ at Layer 1), D an $m \times r$ dictionary matrix with r features, and Z an $r \times n$ coefficient matrix, whose columns contain at most k non-zero coefficients. We use the K-SVD algorithm to solve this problem [10]. K-SVD alternately fixes D to find the best Z via Orthogonal Matching Pursuit (OMP) [11], and then fixes Z to find the best D by a sequence of rank-one approximations. Post training, when dictionaries are fixed, OMP performs feature extraction on new input data.

Max pooling combines the largest components over shifted patches within a window. For example, using three shifted patches, this process would proceed as follows:

1) Given link measurements $x = [x(1), x(2), ...]$, shifted patches of size $m = 3$ are enumerated:

$$X_{\text{win}} = \begin{bmatrix} x(1) & x(2) & x(3) \\ x(2) & x(3) & x(4) \\ x(3) & x(4) & x(5) \end{bmatrix}$$

2) Patches are then sparse coded:

$$Z_{\text{win}} := \min_{Z} \|X_{\text{win}} - DZ\|_2 \quad \text{s.t} \quad \|z_i\|_0 < k$$

with z_i the i^{th} column vector of Z.

3) A pooled feature vector z is computed from the maximum responses over shifted patches:

$$z = \begin{bmatrix} \max(Z_{\text{win}}(1,1), Z_{\text{win}}(1,2), Z_{\text{win}}(1,3)) \\ \vdots \\ \max(Z_{\text{win}}(r,1), Z_{\text{win}}(r,2), Z_{\text{win}}(r,3)) \end{bmatrix}$$

The effect of max pooling is to ensure that extracted feature information is not erroneously altered simply due to artifactual positional shifts.

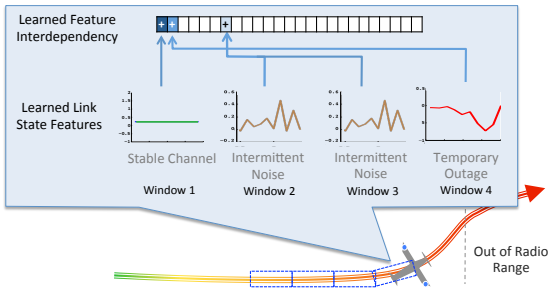

Fig. 3. An example of a learned Layer 2 dictionary feature. The corresponding co-occurring Layer 1 features capture a stable channel that begins to exhibit noise, and intermittent outages. This sequence is associated often with with UAV positions at the edge of radio range.

In our experiments, we found better performance on binary 0/1 packet loss streams than fine-grained SNR data. This can be interpreted using previous arguments by noting that the degree of variation in SNR data is higher than in binary packet reception streams, so latching SNR data to loss/reception "features" acts as an initial sparse coding step that reduces noise to improve classification accuracy.

D. Learned Link Features

To verify that dictionary learning yields meaningful results, we train a simple version of the above model and corrorborate the model's results with experiential knowledge. We set the Layer 1 dictionary size $r_1 = 10$ and sparsity constraint $k_1 = 1$, and the Layer 2 dictionary size $r_2 = 10$ with sparsity $k_2 = 1$. Dictionary learning at Layer 1 will thus find the ten most prominent features in packet reception data, and Layer 2 will find the ten most prominent feature combinations that characterize the link.

Figure 3 illustrates the Layer 2 feature that is most often associated with UAV positions at the edge of radio range under these parameter settings. We see the co-occurrence of three different Layer 1 features, intuitively describing a previously good channel that begins to exhibit intermittent losses, followed by longer outages. This characterization of the UAV link matches with our understanding of intermediate regions of the flight path, where signal strength degrades and causes increased sensitivity to effects like occlusion and antenna nulls that lead to temporary outages.

IV. Adaptive Link Layer Protocol

Our ultimate goal is to use prediction to improve throughput in intermediate regions of the flight path. Furthermore, we would like to support TCP connections, which are ubiquitous due to their reliability and congestion control mechanisms. However, TCP is ill-suited to the changing UAV environment due to its assumption that the transmission medium is stationary. Two major issues result: first, unmitigated link loss is misinterpreted as a congestion signal, causing transmitters to improperly throttle back, and second, when acknowledgements for packets in a transmission window are all lost, TCP must wait for a time out before reviving the connection. Since timeouts are ignorant of the flight path, swaths of close-range

transmission may go unused, or worse, the connection will completely die [23].

In response, we implement an adaptive link layer protocol. The most common method to compensate for intermittent losses is to use a small number of link layer retransmissions [24]. Though the number of retransmissions can be scaled up for especially bad links, this strategy introduces unnecessary channel contention in the network when nodes try repeatedly to send packets over unavailable links. Given the UAV's elevation, these extraneous transmissions could significantly hurt network performance for configurations with additional UAV transmitters or ground links. Instead, using gap prediction, we will queue packets during these outages to avoid TCP performance degradation, and quiet transmitters when appropriate.

To support prediction, we assume a probe that constantly collects link state information. Unlike MIMO systems that must estimate fine-grained link parameters, our prediction requires a coarse 0/1 input stream, and is naturally tolerant to some degree of noise. Therefore, a lightweight side-channel mechanism is sufficient. We earmark the design of an as-light-as-possible probe that exploits our model's tolerance to noisy incomplete data for future work, and use the field-collected packet streams as probe data.

V. Training and Evaluation

A. Training by High Throughput Screening

Previous results have shown that, while the structure of hierarchical inference models drives performance improvements, results are sensitive to parameter settings. Therefore, we adopt high throughput screening [19] to conduct a parallelized sweep of the parameter space, and identify high performing configurations. Models are trained using packet traces from both wing-mounted nodes together, collected over multiple flights on a single day in June. Out-of-range segments, and segments with no loss are excluded from training to focus classifiers on intermediate regions of the flight path. We evaluate performance while scaling up the amount of training data, adding new samples in the order that they appear along the flight path to properly reflect the data acquisition process.

Table I compares precision and recall at each data size using three methods: an SVM applied directly to link data, the best hierarchical inference model configuration identified by screening, and a hybrid model optimized on predictions from both methods simultaneously. This co-optimized hybrid trains both a hierarchical inference model and an SVM on raw data, and logically ORs their predictions to combine complementary results. The resulting prediction accuracies are plotted in Figure 5. We see that the SVM-only method converges quickly to 0.71/0.72 precision and recall on the training set, translating to 78% overall prediction accuracy on the evaluation set. For this method, additional flying time does not improve performance. In contrast, the best trained hierarchical model, with parameters $r_1 = 12, k_1 = 8, r_2 = 40, k_2 = 22$, continues to improve as more data is added, achieving 0.72/0.88 precision/recall, or 84% prediction accuracy. At $51,200$ samples, this represents roughly 30 minutes of flying time for training. We also see that when little training data is available, the two methods complement each other. As a result, the co-optimized hybrid model is useful for bootstrapping link prediction.

TABLE I
Precision / Recall for Gap Prediction on an October Flight

# Training Samples:	400	6400	12800	51200
SVM	0.66 / 0.77	0.70 / 0.74	0.71 / 0.72	0.71 / 0.72
Hierarch. Inf.	0.66 / 0.65	0.72 / 0.68	0.77 / 0.70	0.80 / 0.83
Co-Opt. Hybrid	0.67 / 0.78	0.70 / 0.80	0.72 / 0.81	0.73 / 0.88

While an SVM classifier applied directly to packet traces quickly converges in both precision and recall, our hierarchical model continues to improve classification quality as more training samples are added. Initially, a co-optimized hybrid can boost recall by logically ORing positive alarms from both methods to combine complementary results, however hierarchy's performance eventually surpasses all others.

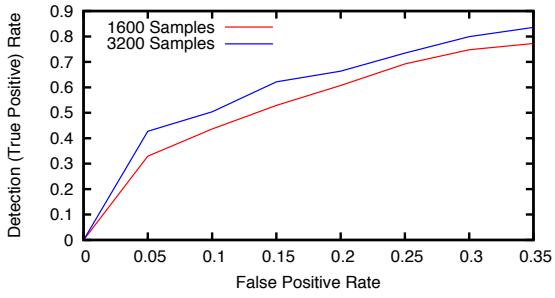

Fig. 4. Receiver Operating Characteristic (ROC) curve, plotting detection rate (true positives) against false positives, for model configurations trained on 1600 samples in red, and 3200 samples in blue. Though multiple parameters are varied within the screening process, these curves represent models with the maximum observed detection rate for a given false positive rate. In order to identify the best model among many configurations, we optimize equally between both metrics. In general, we see that adding training data improves the ROC profile for hierarchical inference.

B. Emulated TCP Results

Finally, we implement our link layer protocol in emulation, replaying the October flight trace. End nodes run live TCP connections, and the emulator delivers or drops packets based on trace data. Upstream ACKs are small relative to the packet

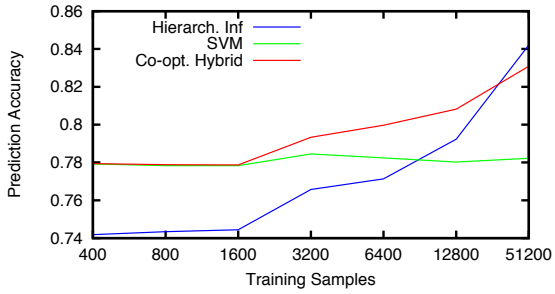

Fig. 5. Prediction accuracy for the three modeling methods in Section V-A on October flight data, plotted logarithmically against the number of training samples. In contrast to the SVM, hierarchical inference improves prediction accuracy as flight time increases. When little training data is available, a hybrid predictor that logically ORs together results from both methods increases accuracy.

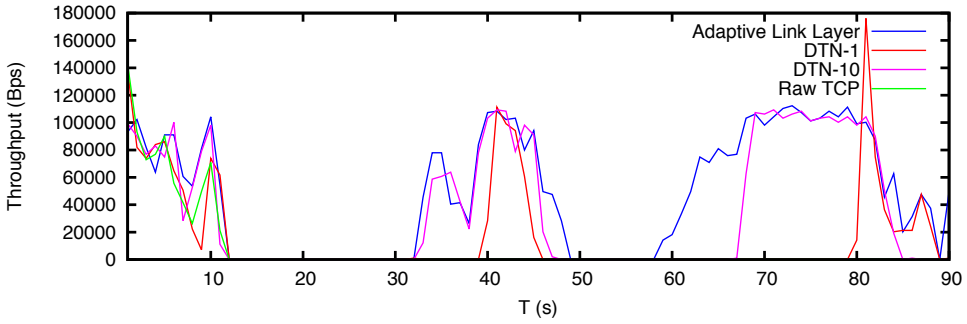

Fig. 6. A comparison of aggregate throughput for one lap of the UAV's flight, using different link layer protocols. In green, raw TCP times out during the first out-of-range portion of the flight, and the connection never resumes. In red, DTN-1 is a Delay Tolerant Networking (DTN) protocol that queues packets when the UAV is out of range (no packet deliveries are possible). This protocol suffers TCP timeouts at the unstable edges of radio range, limiting data transmission to close-range regions of the flight path. In purple, DTN-10 queues packets until 10 packet deliveries are possible, allowing the link to stabilize before resuming transmission. This protocol improves intermediate region transmission at the 30s mark, but utilization is low during the intermediate region at the 60s mark. In blue, our prediction-based adaptive link layer leads to consistently good throughput in intermediate regions.

size in our channel sounding data, so we only modulate data packets, and drop ACKs with 10% probability. Due to the channel's high loss rate, we boost the default retransmissions for all packets to 12 retries, over the Linux default 8.

Figure 6 plots throughput for one lap of the UAV, comparing our adaptive link layer to a raw TCP connection that relies solely on retransmissions, and to two delay-tolerant protocols, DTN-1 and DTN-10 [25]. The DTN protocols queue packets when the UAV is out of range, and rely on retransmissions when the UAV is in range of the ground receiver. DTN-1 detects an in-range transition when a successful packet transmission is first possible, while DTN-10 waits for the link to stabilize enough for ten packets to be transmitted.

In this case, the raw TCP connection sets up properly, but suffers a large throughput drop when the link temporarily destabilizes at the 9s mark. The connection then dies when the UAV passes out of range, and never recovers. The DTN-1 protocol successfully recovers when the UAV comes in range after the 30s mark, however reaction time is slow. This is because many transmissions at the radio range edge fail. This problem occurs again after the 60s mark. Here, several packets are delivered successfully, but are not acknowledged. They are held at the receiver until TCP timeout, when the window is completed, nearly 20 seconds after the link becomes available. DTN-10 fairs better at the 30s mark, and begins quickly, but suffers a delayed reaction after the 60s range transition.

Finally, we see that the adaptive link layer not only prevents out-of-range timeouts, but copes well with radio range edges, improving throughput greatly during these limited opportunities. The total throughput boost is 5.5x over raw TCP, 2.5x over DTN-1, and 1.3x over DTN-10. The comparison difference between DTN-1 and DTN-10 shows how important these intermediate regions are for achieving good throughput. We note that, for this flight path and network configuration, intermediate regions are relatively short in duration. We expect gains from the prediction-based adaptive link layer to increase proportionally to the duration of these intermediate regions in different flight scenarios.

VI. Conclusions & Future Work

In this paper, we applied hierarchical sparse coding and machine learning techniques to implement wireless link prediction for air-to-ground UAV links. This approach computes a non-linear transformation of link data into a sparse variation-tolerant feature space, where prediction is more stable. We show that training samples from flight data can be used for good prediction during future flights in different conditions. Using the resulting predictor, we improve aggregate throughput for TCP connections during regions of intermediate connectivity over the UAV's flight path. To our knowledge, this study is the first to use hierarchical sparse coding for wireless link prediction and TCP performance improvement. Results suggest a promising approach that generalize to a wide variety of scenarios.

Future work will focus on two areas. First, we will conduct additional flight experiments in order to more comprehensively characterize how stable the model's predictive power is across different scenarios. These experiments will include different flight paths, transmitter positions, airspeeds, and ground structure density. Second, we will investigate the minimum requirements for a lightweight channel probe to support inference. This will include measuring the effect on predictive power of probing with smaller packets, possibly on adjacent channels in the frequency space. Furthermore, since inference is naturally tolerant to noisy and incomplete data, we will characterize whether or not opportunistic probes that are derived passively from data transmissions can take the place of a constant active channel probe.

ACKNOWLEDGMENTS

This material is based on research sponsored in part by the Intel Corporation, and by the Air Force Research Laboratory under agreement number FA8750-10-2-0180. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Air Force Research Laboratory, the U.S. Government, or the Intel Corporation.

REFERENCES

[1] J. B. Andersen, T. S. Rappaport, and S. Yoshida, "Propagation measurements and models for wireless communications channels," *Communications Magazine, IEEE*, vol. 33, no. 1, pp. 42–49, 1995.

[2] R. Valenzuela, "A ray tracing approach to predicting indoor wireless transmission," in *Vehicular Technology Conference, 1993., 43rd IEEE*. IEEE, 1993, pp. 214–218.

[3] A. Konrad, B. Y. Zhao, A. D. Joseph, and R. Ludwig, "A markov-based channel model algorithm for wireless networks," *Wireless Networks*, vol. 9, no. 3, pp. 189–199, 2003.

[4] H.-T. Kung, C.-K. Lin, T.-H. Lin, S. J. Tarsa, D. Vlah, D. Hague, M. Muccio, B. Poland, and B. Suter, "A location-dependent runs-and-gaps model for predicting tcp performance over a uav wireless channel," in *Military Communications Conference*. IEEE, 2010, pp. 635–643.

[5] K. Kumar, R. Chandramouli, and K. Subbalakshmi, "On stochastic learning in predictive wireless ARQ," *Wireless Communications and Mobile Computing*, vol. 8, no. 7, pp. 871–883, 2008.

[6] B. Ferris, D. Fox, and N. Lawrence, "Wifi-slam using gaussian process latent variable models," in *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, 2007, pp. 2480–2485.

[7] M. A. Hearst, S. Dumais, E. Osman, J. Platt, and B. Scholkopf, "Support vector machines," *Intelligent Systems and their Applications, IEEE*, vol. 13, no. 4, pp. 18–28, 1998.

[8] V. M. Patel and R. Chellappa, "Sparse representations, compressive sensing and dictionaries for pattern recognition," in *Pattern Recognition (ACPR), 2011 First Asian Conference on*. IEEE, 2011, pp. 325–329.

[9] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005.

[10] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," *CS Technion*, 2008.

[11] Y. C. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition," in *Signals, Systems and Computers, 1993. 1993 Conference Record of The Twenty-Seventh Asilomar Conference on*. IEEE, 1993, pp. 40–44.

[12] N. Pinto, J. J. DiCarlo, and D. D. Cox, "How far can you get with a modern face recognition test set using only simple features?" in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2591–2598.

[13] T. S. Lee and D. Mumford, "Hierarchical bayesian inference in the visual cortex," *JOSA A*, vol. 20, no. 7, pp. 1434–1448, 2003.

[14] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 609–616.

[15] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *Proceedings of the international conference on artificial intelligence and statistics*, vol. 5, no. 2. MIT Press Cambridge, MA, 2009, pp. 448–455.

[16] B. A. Olshausen, D. J. Field *et al.*, "Sparse coding with an overcomplete basis set: A strategy employed by VI?" *Vision research*, vol. 37, no. 23, pp. 3311–3326, 1997.

[17] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 2. IEEE, 2005, pp. 994–1000.

[18] M. J. Gans, N. Amitay, Y. Yeh, T. Damen, R. A. Valenzuela, C. Cheon, and J. Lee, "Propagation measurements for fixed wireless loops (FWL) in a suburban region with foliage and terrain blockages," *Wireless Communications, IEEE Transactions on*, vol. 1, no. 2, pp. 302–310, 2002.

[19] N. Pinto and D. D. Cox, "Beyond Simple Features: A Large-Scale Feature Search Approach to Unconstrained Face Recognition," in *IEEE Automatic Face and Gesture Recognition*, 2011.

[20] M. Norouzi, M. Ranjbar, and G. Mori, "Stacks of convolutional restricted boltzmann machines for shift-invariant feature learning," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2735–2742.

[21] M. Steyvers and T. Griffiths, "Probabilistic topic models," *Handbook of latent semantic analysis*, vol. 427, no. 7, pp. 424–440, 2007.

[22] N. Pinto, D. D. Cox, and J. J. DiCarlo, "Why is real-world visual object recognition hard?" *PLoS computational biology*, vol. 4, no. 1, p. e27, 2008.

[23] C.-K. Lin, H.-T. Kung, T.-H. Lin, S. J. Tarsa, and D. Vlah, "Achieving high throughput ground-to-uav transport via parallel links," in *Computer Communications and Networks, 2011 Proceedings of 20th International Conference on*. IEEE, 2011, pp. 1–7.

[24] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving tcp performance over wireless links," *Networking, IEEE/ACM Transactions on*, vol. 5, no. 6, pp. 756–769, 1997.

[25] C.-M. Cheng, P.-H. Hsiao, H.-T. Kung, and D. Vlah, "Maximizing throughput of uav-relaying networks with the load-carry-and-deliver paradigm," in *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*. IEEE, 2007, pp. 4417–4424.