Online Feature Selection for Semantic Image Segmentation

Rishav Rajendra
Canizaro Livingston Gulf States
Center for Environmental Informatics,
New Orleans, USA
email: rrajendr@uno.edu

Chris J. Michael
Naval Research Laboratory
Center Geospatial Sciences
Stennis Space Center
Mississippi, United States
email: chris.michael@nrlssc.navy.mil

Elias Ioup
Naval Research Laboratory
Center Geospatial Sciences
Stennis Space Center
Mississippi, United States
email: elias.ioup@nrlssc.navy.mil

Md Tamjidul Hoque
Canizaro Livingston Gulf States Center for Environmental Informatics,
New Orleans, USA
email: thoque@uno.edu

Mahdi Abdelguerfi
Canizaro Livingston Gulf States Center for Environmental Informatics,
New Orleans, USA
email: mabdelgu@uno.edu

Abstract— **In this project, we classify each pixel from the incoming stream of aerial imagery of water bodies as either "land" or "water" in real-time. Traditional batch feature processing techniques can be too slow to adapt to real-time changes. This paper proposes an online distributed framework for Semantic Segmentation using conditional independence to discard irrelevant and redundant features to train a fast and lightweight but accurate machine learning model. Through extensive experimental results using aerial imagery of water bodies, we demonstrate that our approach is faster than existing online feature selection methods while maintaining high accuracy.**

Keywords - machine learning; semantic segmentation; streaming images; feature selection.

I. Introduction

Traditional feature selection algorithms require all data-points to be available and presented before the feature selection process starts [6]. After all the features have been collected, the feature selection process begins. This is not always possible in the real world because we do not always know where the end-point is. In this work, we aim to solve this problem in terms of streaming aerial images of water bodies. As images arrive, we generate candidate features dynamically one at a time. We believe generating features one at a time provides a greater practical advantage over traditional feature selection. For example, in this research, we classify each pixel in an aerial image of water bodies into two classes: land and water. A single channel from the smallest image in our dataset contains 468,784 pixels (706 weight × 664 height). The images we use have four channels per image. As a result, the computational cost of generating features from these images is high. We believe waiting for the feature extracting process to complete before the learning begins is not practical for a real-time use case. It is preferable to generate features one at a time [10]. Online feature selection seeks to select the minimal set of features from the incoming features as they arrive while maintaining a high overall model accuracy. Online feature selection stores all the incoming data in two primary data structures: streaming data structure and streaming features structure.

A preliminary distinction is needed between streaming data structure and streaming features structures. For streaming data structure, the number of features selected remains the same throughout the entire feature selection process. Still, the number of data instances increases over time. However, in streaming features structure, the number of data instances per feature is fixed, but the number of features increases over time.

Let us assume an algorithm chooses five features for both streaming data structure and streaming features structure after the first image. Assuming the number of features remains constant for streaming data structure but increases by one for every subsequent image in streaming features, even if the total number of features selected remains the same with streaming data structure, the total number of data instances across all five features will always increase after every image. However, for streaming features structure, even if we select five features after the first image and an additional feature with every new image, we will have significantly fewer data instances over time. Streaming features structure will only be a problem if our feature selection algorithm selects a very high number of features from every image. Ideally, our feature selection algorithm will choose only a small number of features giving streaming feature structure a significant advantage.

Two notable research efforts have greatly contributed to addressing the problem of online feature selection using a streaming feature structure. Zhou *et al.* presented Alpha-investing [15] for streaming feature selection. With Alpha-investing, Zhou *et al.* mainly focused on controlling the threshold during feature selection. Alpha-investing uses a p-value from linear and logistic regression to dynamically adjust the threshold while selecting new features. Alpha being "invested" increases the wealth and threshold to allow for a slight increase in the inclusion of incorrect features. However, for every instance when a feature from the dynamically generated stream is tested to be insignificant, wealth is "spent" which reduces the threshold. Alpha-investing can handle an infinite number of features, but only evaluates each new candidate feature exactly once without considering the redundancy of the selected features. On highly redundant datasets like the one we are using, Alpha-investing provides a very low and unstable prediction accuracy.

Koller *et al*. proposed a classification of input features F with respect to their relevance to a target T in terms of conditional independence [5][6]. They propose a learner-independent paradigm for feature subset selection, viewing an induction algorithm as a biased method for approximating the probability distribution of class labels given features and transforming this distribution of class labels that the induction algorithm attempts to approximate. They also classify elements into three disjoint categories belonging to X input features and their importance in C target class: (1) strongly relevant, (2) weakly relevant, and (3) irrelevant. Yu and Liu [13] improved this categorization by proposing a definition of feature redundancy, therefore, creating a path for efficient elimination of redundant features. For the following definitions, let F be a full set of features, F_i denotes the ith input feature, C denotes the target, and $S = F - \{F_i\}$ represent all input features excluding F_i.

Definition 1 (Conditional Independence) In a feature set F, two features F_i and F_j are conditionally independent given the set of features Z, if and only if

$P(F_i|F_j,Z) = P(F_i|Z)$, denoted as Independent$(F_i,F_j|Z)$.

Definition 2 (Strong relevance) Feature F_i is strongly relevant to C if and only if

$$P(C|F_iS_i) \neq P(C|S_i).$$

Definition 3 (Weak relevance) Feature F_i is weakly relevant to C if and only if

$P(C|F_i,S_i) = P(C|S_i)$, and $\exists S_i \subset S_i$, such that $P(C|F_i, S_i) \neq P(C|S_i)$.

A feature with weak relevance is not always in the final, optimal feature subset, but ideally, it would be included.

Definition 4 (Irrelevance) Feature F_i is irrelevant to C if and only if

$$\forall S \subseteq S_i, P(C|F_i,S_i) = P(C|S_i).$$

Yu and Liu [13] proposed dividing features into necessary and unnecessary features. In their definition derived from the Markov blanket, redundant features provide no additional information than features already selected, and irrelevant features provide no useful information in the final model.

Definition 5 (Markov blanket) Given a feature F_i, let $M_i \subset F(F_i \notin M_i)$, M_i is said to be a Markov blanket for F_i if and only if

$$P(F - M_i - \{F_i\}, C|F_i, M_i) = P(F - M_i - \{F_i\}, C|M_i).$$

where C is the Markov blanket. We can eliminate conditionally independent features from the selected candidate feature set using the Markov blanket without increasing the distance from the desired distribution [8].

Definition 6 (Redundant feature) Let G be the current set of features. A feature is redundant and hence needs to be removed from G if and only if there is a weak relevance and has a Markov blanket M_i within G.

In another study, Wu *et al*. developed a new framework that used feature relevance and a new algorithm called Online Streaming Feature Selection (OSFS) [12]. OSFS uses a two-step approach to discard irrelevant and redundant features from the streaming features as they arrive. Based on the definitions above, the entire feature set is divided into four basic disjoint parts: (1) irrelevant features, (2) redundant features, (3) weakly relevant but non-redundant features, and (4) strongly relevant features. First, the framework conducts an online relevance analysis, Definition 4, which determines a new feature with respect to its relevance to the target T and removes irrelevant ones. After that, the online redundancy analysis, Definition 6, eliminates redundant features from the features selected so far. These two steps are repeated one after the other until a stopping criterion is satisfied.

In Section 2, we go through the dataset we used to test our proposed frameworks, image augmentation and feature extractions methods used, the online feature selection framework used in this paper and the performance evaluation metrics used to judge models. In Sections 3 and 4, we discuss the results of proposed frameworks and the conclusion respectively.

II. Methods

In this section, we describe the approach for benchmark and independent test data preparation, feature extraction, performance evaluation metrics, and finally, the path we took to establish the feature selection framework for semantic image segmentation.

A. Dataset

The images used for this work are aerial imagery of water bodies acquired during the agricultural growing seasons in the continental US by the National Agriculture Imagery Program (NAIP). NAIP is administered by the USDA's Farm Service Agency (FSA) through the Aerial Photography Field Office in Salt Lake City. This "leaf-on" imagery is used as a base layer for GIS programs in FSA's county service centers and is used to maintain the Common Land Unit (CLU) boundaries.

NAIP imagery is acquired at a one-meter Ground Sample Distance (GSD) with a horizontal accuracy that matches within six meters of photo-identifiable ground control points, which are used during image inspection.

We have a total of eight images. All images are captured with four bands of data: red, green, blue, and near-infrared. Every picture complies with the specification of no more than 10 percent cloud cover per quarter quad tile, weather conditions permitting. All imagery is inspected for horizontal accuracy and tonal quality.

B. Image Augmentation

As we have a deficient number of images, eight, we used various image augmentation techniques to increase the size of the available dataset. A total of eight image augmentation methods were used on each image channel.

The following image augmentation methods were applied on each image channel in the dataset mentioned in 2.A:

a) Random Image Rotation Augmentation

b) Random Flip Augmentation

c) Random Shift Augmentation

d) Random Channel Shift Augmentation

e) Gray Scale

f) Random Brightness Adjustment

g) Random Contrast Adjustment

C. Feature Extraction

Feature extraction can provide new attributes. After each image arrives and image augmentation methods are applied, features are extracted dynamically from each augmented image. Extracted features are then sent to the online feature selection framework. We derive three main features from the images: Gabor Kernel Features, Canny Edge Detector, and Gaussian Blur.

a) Gabor Kernel Features: Gabor Kernel Features are special classes of bandpass filters, i.e., they allow a specific 'band' of frequencies and reject the others. Gabor kernel-based features have been successfully and widely applied to a broad range of image processing tasks like texture recognition and face recognition [11]. This is because the characteristics of the Gabor kernel, mainly the frequency and orientation representations, are similar to those of the human visual system [7]. We extracted the Gabor features based on five parameters: (1) λ - Wavelength of the sinusoidal component, (2) θ - The orientation of the normal to the parallel stripes of the Gabor function, (3) ψ - The phase offset of the sinusoidal function, (4) σ - The standard deviation of the Gaussian envelope and (5) γ - The spatial aspect ratio and specifies the ellipticity of the support of the Gabor function. These five parameters control the shape and size of the Gabor function.

b) Canny Edge Detector: Canny Edge Detection is widely used in computer vision to locate sharp intensity changes and to locate object boundaries in an image [3]. A Canny Edge Detector classifies a pixel as an edge if the gradient magnitude of the pixel is more significant than those of pixels at both sides in the direction of maximum intensity change. It is optimal, according to the three criteria of proper detection, sound localization, and a single response to an edge [3]. We extracted the features from Canny Edge Detection using the OpenCV's implementation of the Canny Edge Detection algorithm [1]. The feature extraction process goes through different stages like Noise Reduction, finding the intensity gradient of the image, Non-maximum suppression, and Hysteresis thresholding.

c) Gaussian Blur: Gaussian Blur reduces the noise and detail of the image. This algorithm is applied to provide our frameworks "bad" or distorted data to create a robust model that can be reliable and used in real-life environments.

D. Online Feature Selection Framework

For real-time semantic image segmentation, we propose a new framework that accepts an image stream, applies the image augmentation techniques, extracts features from the images, and discards irrelevant and redundant features automatically. Due to the highly redundant images in our dataset, we believe the online relevancy analysis and online redundancy analysis from the OSFS framework [12] will be able to select the least number of features from the entire feature set while maintaining high and stable accuracy.

From a cold-start, OSSF initializes an empty feature set. After an image is presented to the OSSF, all four channels in the image are augmented one by one. Upon completion of the augmentation, OSSF extracts the feature vectors from the augmented image. While all features from the augmented image have not processed, check if the feature is relevant or not. If the feature is relevant, conduct the redundancy analysis. If the relevant feature is not redundant with other selected features, add the feature to the feature set. Process the next feature from the augmented image. After all features of that augmented image have been processed one-by-one, move on to the next augmented image. After processing all the augmented images, move on to the next channel of the original image. After all, channels have been processed, move to the next image from the data source until there are no images, or a pre-set condition is reached.

Algorithm 1: Online Feature Selection for Semantic Image Segmentation

```
input  : image_stream
output: Best candidate features (BCF)
BCF = [];
while image_stream do
    for channels in image do
        // Augment each incoming channel
        for augmented channel do
            while GetNextFeature do
                $F_i$ ← GetNextFeature();
                // Online relevance analysis
                if $F_i$ relevant then
                    BCF = BCF ∪ $F_i$
                    // Online redundancy analysis
                    for each feature $Y \in BCF$ do
                        if $\exists S \subseteq BCF \setminus Y$, s.t. $Ind(Y, C \mid S)$ then
                            BCF = BCF - Y;
                        end
                    end
                end
            end
        end
    end
end
```

Figure 1. The Online Semantic Segmentation Framework (OSSF).

We implemented our proposed framework, Figure 1, in Python for testing. To determine a given features' conditional independence, we used SciPy's implementation of the chi-square test of independence. The chi-square of independence is used to determine if there is a significant relationship between the features. Null-hypothesis of the chi-square test is that there is no association between the features. For the hypothesis test for the chi-square test of independence, the test statistic is computed and compared to a critical value. The critical value of the chi-square statistic is determined by the level of significance, typically 0.05, and the degrees of freedom. If the chi-square test statistic is higher than the critical value, the null-hypothesis is rejected, and the features are classified as conditionally independent. In the redundancy analysis phase, we check if there is a subset of features from the features selected so far, which is conditionally independent of the class label. We again use

SciPy's chi-square test implementation described above for this functionality. If it is independent, then those features are classified as redundant and discarded.

We believe our framework can be significantly improved using a distributed approach. In a distributed environment, data and jobs are divided across multiple clusters by a driver program. A cluster is a group of computers that work together essentially as a single system. In OSSF, after one image arrives, we need to process all four of its channels sequentially. Each channel produces multiple augmented channels, and each augmented channel produces many features. Until all the features of the augmented channel have been produced one-by-one, the entire framework is suspended before moving to the next augmented channel. Even if a new image has already been presented to the framework, it must wait for the previous image to finish processing. This is very inefficient if the data-source is sending images at a fast rate. To tackle this problem, we propose D-OSSF, a distributed version of our framework where images are processed as soon as they arrive concurrently.

For D-OSSF, we use a Kafka producer to send images to a Spark Streaming client. We designed the Kafka producer to submit a new image to the Spark Streaming client every two seconds. The Spark Streaming client loads in the images using Spark's built-in image source API into Resilient Distributed Datasets (RDD). As the images continue to come in, Spark Streaming client creates a continuous series of RDDs, also known as a DStream. Each RDD in a DStream contains images from certain intervals. The Spark engine then transforms the DStream based on our online feature selection framework [14]. The Spark engine handles the underlying distribution operations on the DStreams and provides a high-level API for convenience.

E. *Performance Evaluation*

To evaluate the performance of our framework, we adopted a widely used 10-fold Cross-Validation (CV) approach. In the process of 10-fold CV, the dataset is segmented into ten parts. When one fold is kept aside for testing, the remaining nine folds are used to train the classifier. This process of training and test is repeated until each fold has been kept aside once for testing, and consequently, the test accuracies of each fold are combined to compute the average [4]. AUC is the area under the Receiver Operating Characteristics (ROC) curve, which is used to evaluate how well a predictor separates two classes of information (land and water in images). We used all the performance evaluation metrics listed in Table 1 below, as well as ROC and AUC, to test the performance of the proposed framework and test it with the existing approaches.

TABLE I. NAME AND DEFINITION OF THE EVALUATION METRIC

Name of Metric	**Definition**	**Formula**
Accuracy (ACC)	The ratio of samples predicted correctly out of the total sample.	$\frac{TP+TN}{FP+TP+TN+FN}$
Balanced Accuracy (BACC)	Average of recall and specificity.	$\frac{1}{2}\left(\frac{TP}{TP+FN}+\frac{TN}{TN+FP}\right)$
Precision (PR)	The ability of the classifier to not label a negative sample positive.	$\frac{TP}{TP+FP}$
Average Precision (AP)	Combines recall and precision for ranked retrieval results.	$\sum_{n}(R_n - R_{n-1})P_n$
Recall	Ability of the classifier classifying positive samples.	$\frac{TP}{TP+FN}$
F1 Score	Harmonic mean of Precision and Recall.	$\frac{2TP}{2TP+FP+FN}$

For all definitions in Table 1, let TP be the number of true positives, TN be the number of true negatives, FP be the number of false positives, FN is the number of false negatives, and P_n and R_n be the precision and recall at the nth threshold respectively.

To test our sequential framework, we use default Scikit-learn's implementation of Logistic Regression (LR), Random Forest (RF), and Decision Tree (DT) classifiers [9]. We also use the eXtreme Gradient Boosting (XGBC) classifier by the Distributed (Deep) Machine Learning Community (DMLC) group [2]. For our distributed framework, we used default Logistic Regression, Random Forest, and Decision Tree classifiers from Spark's MLlib machine learning library [14].

III. RESULTS

In this section, we demonstrate the results of our sequential and distributed frameworks. We also compare our frameworks with Alpha-investing as a benchmark. All experiments were conducted on a computer with two AMD Opteron™ Processor 4386 (3.1 GHz) and 62 GB RAM. All frameworks were given the same sequence of images to avoid any bias. All the experiments were run a total of five times and averaged to minimize inconsistencies.

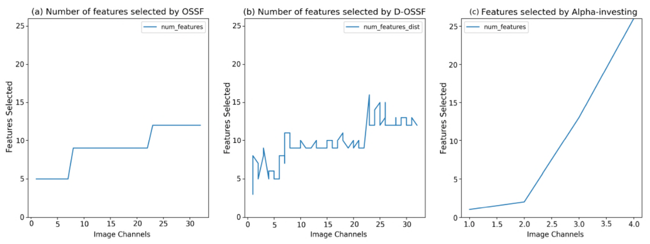

Figure 2. The number of features selected as image channels increase.

As seen in Figures 2a and 2b, both OSSF and D-OSSF end up picking twelve features after the feature selection process completed. We have eight images in our dataset and four channels per image. Every channel leads to nine augmented channels, and every augmented channel generates 32 features. So, both our frameworks, OSSF and D-OSSF, select a subset of 12 features out of 9,216 available features (8 images * 4 channels * 9 augmented channels * 32 features), thus discarding 99.87% of the incoming features.

Figure 2a also shows that the number of features selected in the OSSF is exceedingly stable compared to D-OSSF in Figure 2b. OSSF is more stable because channels are processed one at a time. Features from a new channel are only processed after the current channel has been fully processed. So, the framework gets to run the online relevancy analysis and the online redundancy analysis before returning the selected feature set. In the distributed framework, channels are processed concurrently. Features than are passed to the online relevancy analysis in one of the executors of the Spark ecosystem are added into the candidate feature set and may not be able to go through the redundancy test before another executor returns the feature set. This is a classic example of concurrency where multiple operations are happening at once. But this is not a problem as the selected feature set goes through the redundancy analysis eventually and discard the redundant features. This is proved as the number of features selected at the end of the process across multiple runs in both algorithms is equal. However, the number of features chosen by Alpha-investing, Figure 2c, goes up rapidly as the number of image channels increases. This caused a memory overflow across multiple runs, and we could not process the entire dataset due to hardware limitations. The overflow in Alpha-investing usually occurs in highly redundant datasets like the one we are using as it does not conduct a redundancy analysis.

TABLE II. EVALUATION METRICS OF OSSF (IN %)

Model	ACC	PR	BACC	AP	Recall	F1 Score
DT	91.68	91.39	90.72	92.34	95.63	93.27
LR	91.71	91.39	90.73	92.16	95.72	93.30
RF	91.68	91.39	90.73	92.33	95.63	93.27
XGBC	91.71	91.39	90.72	92.34	95.63	93.27

The ACC of OSSF increases from 86.63% after the first channel to 91.68% at the end for DT, RF and XGBC, a 5.51% increase. Similarly, the ACC of LR increases from 83.23% to 91.91%, a 9.25% increase. Other metrics follow a similar trend. For DT, PR increases from 88.20% to 91.39%, BACC increases from 86.06% to 90.72%, AP increases from 89.54% to 92.34%, Recall improves from 89.36% to 95.63%, and F1 Score increases from 88.63% to 93.27%. We can see that the models learn and improve over time as they gets more data. The performance of Alpha-investing was very erractic with some models even reaching 0% for PR, AP and Recall. Overall, Alpha-investing the metrics for Alpha-investing started pretty high but sharply decreased as the number of image channels increased.

TABLE III. EVALUATION METRICS OF D-OSSF (IN %)

Model	ACC	PR	BACC	AP	Recall	F1 Score
DT	90.45	90.72	90.45	90.25	95.25	94.82
LR	91.39	90.93	91.39	90.27	93.97	93.22
RF	89.95	90.72	89.95	91.01	94.87	93.22
XGBC	91.38	90.72	91.38	92.35	93.51	94.82

Evaluation metrics of D-OSSF, Table 3, follow similar trends to OSSF results. The ACC of D-OSSF increases from 85.43%, 84.11%, 87.43%, and 86.49% to 90.67%, 91.23%, 89.73%, and 91.24% for DT, LR, RF and XGBC respectively. That is a 5.78%, 7.85%, 4.17%, and 5.20% increase respectively. For DT, PR increases from 88.73% to 90.72%, BACC increases from 85.39% to 90.45%, AP increases from 87.98% to 90.25%, Recall improves from 89.05% to 95.25%, and F1 Score increases from 86.63% to 94.82%. D-OSSF's distributed framework does not degrade the performance of models and follows the performance of OSSF very closely.

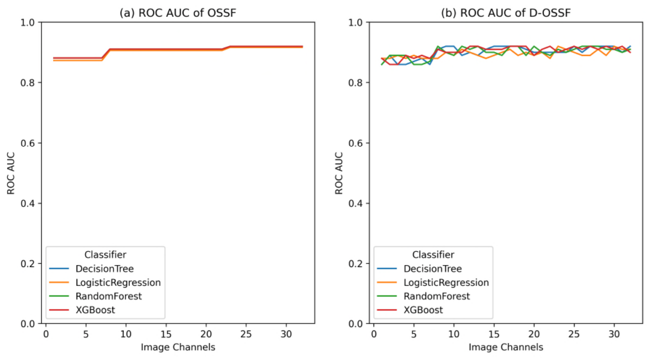

Figure 3. ROC AUC of all three frameworks.

From Figure 3a and 3b, we can see that both our sequential and distributed frameworks achieve comparable results. The AUC of ROC of OSSF, Figure 3a, goes from 0.88 after the first channel to 0.92 at the end for XGBoost, Random Forest, and Decision Tree classifiers, a 4.16% increase. The AUC of ROC of Logistic Regression goes from 0.87 after the first channel to 0.92 by the end, a 5.06% increase. Similarly, the ROC AUC of Decision Tree, Logistic Regression, Random Forest and XGBoost in D-OSSF, Figure 3b, go from 0.86, 0.88, 0.86, and 0.88 to 0.92, 0.91, 0.91 and 0.90 respectively.

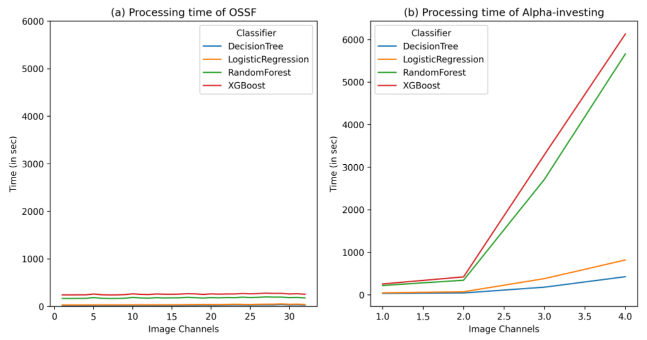

Figure 4. Comparison between the run-time of our OSSF and Alpha-investing.

From Figure 4b, we can see that the erratic nature of Alpha-investing finally ends with the time taken to process each channel going up rapidly before crashing on the fourth channel. The processing time of OSSF, Figure 4a, remains relatively constant as the number of channels increases.

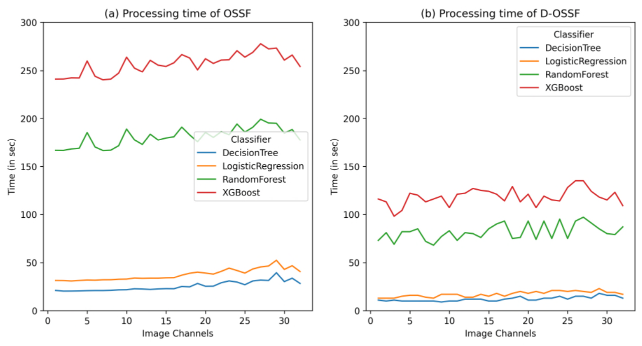

Figure 5. Comparison between the run-time of OSSF and D-OSSF.

From Figure 5a, we can see that for OSSF, on average, the Decision Tree classifier takes the least amount of time with 25.53 seconds. Logistic regression is next on the line with an average of 37.50 seconds. Random Forest takes 180.80 seconds, and XGBoost takes 257.00 seconds on average. From Figure 5b, we observe that for D-OSSF, the Decision Tree classifier again takes the least amount of time with just 12.13 seconds on average, and Logistic Regression takes 17.22 seconds. The Random Forest and XGBoost take 81.81 seconds and 118.34 seconds on average, respectively. D-OSSF, on average, decreases the overall time taken by almost 54% across all classifiers.

IV. CONCLUSIONS

Our aim with this project was to create a framework for Online Semantic Segmentation, which takes in images on the go, extracts, and selects a very low number of features while maintaining a high of model accuracy in real-time. These frameworks are especially important in unknown real-life environments where we do not have previous knowledge of the subject and images stream in as time progresses.

A. Summary

In this research work, two novel frameworks have been developed. These two frameworks are summarized below:

a) Sequential Online Feature Selection Framework: We developed a novel sequential framework for Online Semantic Segmentation that accepts images one at a time, extracts, and selects features on the go. This framework's final accuracy of 91.39% and average processing time per image channel of 25.53 seconds with Decision Tree classifier outperforms other online feature selection algorithms. The 5.51% increase in accuracy over time also proves that our framework can improve as the size of our dataset increases.

b) Distributed Online Feature Selection Framework: Using a distributed Spark ecosystem, we reduced the overall run-time of our framework by almost 54% across all classifiers. The distributed framework produces almost exactly the same performance metrics and selects the same number of features. With the final accuracy of 92.17% and average processing time per image channel of just 12.13 seconds with Decision Tree classifier, we believe our model can be used for real-time implementations.

B. Future Scopes

The frameworks proposed in this research are novel approaches to online Semantic Segmentation by extracting and selecting features on the go. As the size of datasets grows, the importance of online feature selection will grow. The methods used in this research work can also be applied to other fields of computer vision, which require fast training and deployment. We hope to inspire new research in the area of distributed online feature selection across diverse fields.

REFERENCES

[1] G. Bradski and A. Kaehler, "Learning OpenCV: Computer Vision With The OpenCV Library," O'Reilly Media, Inc, 2008.

[2] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," Proceedings of the 22nd ACM Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785-794.

[3] L. Ding and A. Goshtasby, "On The Canny Edge Detector," Pattern Recognition 2001, pp. 721-725.

[4] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning," Springer Series in Statics, 2009.

[5] G. H. Jogn, R. Kohavi, and K. Pfleger, "Irrelevant Features and the Subset Selection Problem," Machine Learning Proceedings, 1994, pp. 121-129.

[6] D. Koller and M. Sahami, "Toward Optimal Feature Selection," Stanford InfoLab, 1996.

[7] C. Lee and S. Wang, "Fingerprint Feature Extraction Using Gabor Filters," Electronics Letters, 1999, pp. 288-290.

[8] J. Pearl, "Probabilistic Reasoning in Intelligent Systems", Networks of Plausible Inference, 2014.

[9] F. Pedregosa *et al.* "Scikit-learn: Machine Learning in Python," The Journal of Machine Learning Research 12, 2011, pp. 2825-2830.

[10] S. Perkings and J. Theiler, "Online Feature Selection Using Grafting," International Conference on Machine Learning, 2003, pp. 592-599.

[11] T. Weldon, W. Higgins, and D. Dunn, "Efficient Gabor Filter Design For Texture Segmentation," Pattern Recognition, 1996, pp. 2005-2015.

[12] X. Wu, K. Yu, H. Wang, and W. Ding, "Online Streaming Feature Selection," International Conference on Machine Learning, 2010, pp. 1159-1166.

[13] L. Yu and H. Liu. "Efficient Feature Selection Via Analysis Of Relevance And Redundancy," Journal of Machine Learning Research, 2004, pp. 1205-1224.

[14] M. Zaharia *et al.* "Apache Spark: A Unified Engine For Big Data Processing," Communications of the ACM, 2016, pp. 56-65.

[15] J. Zhou, D. Foster, R. Stine, and L. Ungar, "Streaming Feature Selection Using Alpha-Investing," Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, 2005, pp. 384-393.