21, rue d'Artois, F-75008 PARIS
http : //www.cigre.org

CIGRE US National Committee
2018 Grid of the Future Symposium

Validating a Google TensorFlow Image Classifier

B. LOGALBO
Leidos
USA

SUMMARY

Utilities rarely have their inventory of assets either completely recorded or completely described. Conflating is the act of attempting to fully complete all descriptive aspects of an asset including taking a full inventory of their assets. Methods of conflation include taking all known sources of information which have the potential of completing any one or more descriptive aspect of an asset already known or can potentially include an asset owned by the utility but whose record is not yet part of the utility's inventory. Google Streetview is a potential source of information which may include the utility's assets not yet included in the utility's inventory record. Further Google Streetview may include location information that is incorrect or not actually entered in the given's asset's location record. However other sources of images can replace the Google Streetview, if more recent than the Google Streetview image – the Google Streetview image can serve as the default image for classification if it is the latest known image available. This paper shall assume that the Google Streetview image is the latest image available. The paper shall assume that if no other sufficient information can be substantiated to conflate the asset record, that the Google Streetview image is last know resource available from which a record of the asset can be conflated.

This document is an overview describing a data science validation approach (K-Fold Cross-validation) applied to the output of a Leidos Digital Utility Google TensorFlow classifier, a classifier which has the ability to classify utility pole equipment from Google Streetview images. This overview describes a method of applying K-Fold Cross-validation when the available ground truth (GT) sets used for validation have a disproportionate distribution of classes and class combinations. If the class distribution is not accounted for appropriately in the validation process, especially for disproportionate distribution of class combinations in image validation sets, the performance of the classifier will be misrepresented.

Typically applications are delivered to customers under contractual performance requirements; the consequences of misrepresenting contractually specified image classification performance can potentially result in significant unanticipated cost to both the customer and service provider. For example, the customer must accommodate the effects of the application's unanticipated poor performance on their day-to-day operations; the service

Robert.D.LoGalbo@leidos.com

provider, who failed to meet the performance terms of their contract, must account for breach of contract or prepare to pay a penalty.

Stratification sampling has been discussed in literature in generally non-specific terms to account for this validation environment. This paper introduces an extended and inclusive method to eliminate selection bias, rendering a full understanding of classifier performance for this commonly encountered validation environment.

KEYWORDS

k-fold cross-validation, stratification sampling, selection bias, training, TensorFlow, classification, deep learning, machine learning, precision, recall

Assumptions

The underpinnings of Google TensorFlow (TF) will not be discussed here and this information is readily available in the public domain. The TensorFlow classifier is a Deep Learning (DL) classifier which can be applied as an image classifier, made freely available by Google in late 2015, slightly over 2 years ago[1]. When trained, the TF classifier can accept an image and decide whether or not any of the classes upon which it was trained are present in the given image. From a test set the TF classifier can report, with detection scores: True Positives (TP), False Positives (FP), and False Negatives (FN).

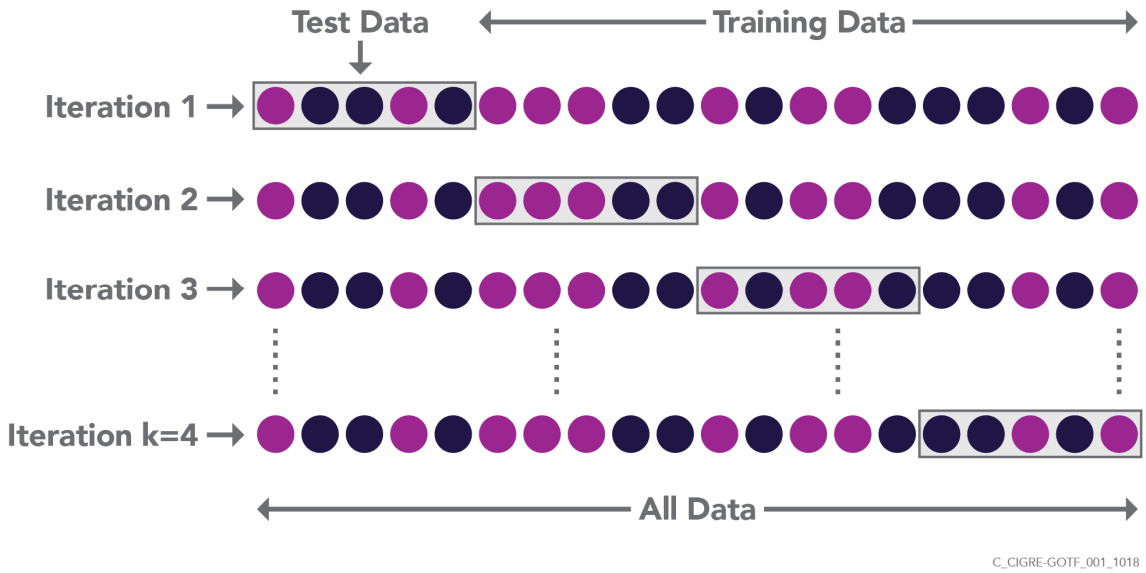
Google TensorFlow Application Applied to Utility Pole Equipment

For this application, TF has been trained on thousands of images containing Transformers (T), Streetlights (S), fuses (F), miscellaneous equipment (telecommunication "boxes," capacitor banks, etc., abbreviated by the letter C) and images without any relevant classes (termed E for empty) (i.e., there are L=5 classes). TF must be appropriately configured, customized, and trained with samples from these classes to perform well. Once this process is complete, TF can detect with very high precision, recall (i.e., sensitivity), and F1 scores (i.e., the harmonic mean of the precision and recall), any of the above classes with scores of about 90%. The difficulty is understanding how to derive those values with a confidence that can be reported to a customer and trusted to scope project cost.

Accuracy is a poor metric for this application given that the set of True Negatives (TN) is so large. The value in understanding this decision algorithm (i.e., classifier) is to understand its precision, recall, and F1 score. As a brief reminder, precision is (TP/[TP+FP]) and recall is (TP/[TP+FN]). Therefore, if an image with no transformer was classified as having one, that test would qualify as an FP. If an image had a transformer and was classified as not having one, that test would be considered an FN. Evaluating precision, recall, and F1 scores is based upon the process of counting TPs, FPs, and FNs.

The Gold Standard of Performance Validation: K-Fold Cross-validation

K-Fold Cross-validation, in the data science community today, is colloquially considered the "gold standard" of validation[2]. Its benefits come from the fact that all image samples are eventually used in the training process and all image samples are eventually used in the testing process. This validation process tends to reduce the variance of performance metrics and the probability of overtraining (increased performance variance when implemented for a customer) given that there are multiple sets (i.e., K sets for binary classification) of training. A visual representation of K-fold cross-validation is depicted below with four iterations of testing (K=4). In each of the four iterations depicted below, the "box" containing five samples is the test set for the specific iteration; the remaining samples not in the "box" in the specific iteration are used for training. Note that each sample is in a test set exactly once, and each sample is in a training set in all other iterations (in this example, three times):



The process of K-fold Cross-validation will be explained using the following example[3]. In the following example, k=10 (given it is mathematically more convenient than 4) and for a binary classification, a one-vs-all classifier.

Assume we have N=2500 labeled GT images. We randomly select 90% of the 2500 labeled GT images to train and 10% to test (i.e., 250 randomly selected for test and the 2250 that remain for training). After running the test with the 250 randomly selected images used for testing, we perform the counting of TP/FN/FP values as discussed above, generating the first table of counting results for the first fold. (The term "fold" refers to a validation iteration.)

We repeat this process but in each subsequent iteration, the 10% that was left out for validating performance are included in the training set for the next run (i.e., the next fold), and 10% of the data that had been used to train are randomly selected to become the test data for the next fold. This continues until every image has had a chance to be in the training set and every image has had a chance to be in the test set; therefore, in our example this is run 10 times (i.e., 10 folds) because k=10.[1]

Continuing with the example for how to execute the second fold, 250 samples would be randomly selected from the 2250 GT images used for training in the first fold. The 250 samples used for testing in the first fold are now added into the remaining 2000 images for training (i.e., 250 images from the first test set + 2000 images puts us back to 2250 to be used for training in the second fold). Again, the 250 images randomly selected for testing in the second fold are run through the classifier. Counting is performed a second time on the output. A second accounting of TP/FN/FP values is generated for the second fold.

Continuing with this example for the third fold, 500 images are now reserved for training (i.e., the 250 images used for testing from the first fold and the 250 images used for testing in the second fold). We now randomly select 250 images from the remaining 2000 images for the test set. This leaves 1750 images, plus the 250 images from the testing in the first fold, plus

[1] If k=8, this validation would be run 8 times (i.e., 8 folds) and instead of selecting 10%/90% (testing/training) per fold, it would be 12.5%/87.5% per fold.

the 250 images used for testing in the second fold, putting us back to 2250 images to be used for training. After running the 250 images randomly selected for testing from the third fold through the classifier, counting is performed a third time on the output. A third counting of TP/FN/FP values is generated for the third fold.

Continuing with this example for the fourth fold, 750 images are now reserved for training (i.e., the 250 images used for testing from the first fold, the 250 images used for testing in the second fold, and the 250 images used for testing in the third fold). We now randomly select 250 images from the remaining 1750 images for the test set. This leaves 1500 images, plus the 250 images from the testing in the first fold, plus the 250 images used for testing in the second fold, plus the 250 images from the testing in the third fold, putting us back to 2250 images to be used for training. After running the 250 images randomly selected from the fourth fold through the classifier, counting is performed a fourth time on the output. A fourth counting of TP/FN/FP values is generated for the fourth fold.

This process continues accordingly for the fifth, sixth, seventh, eighth, and ninth folds.

On the tenth and final fold there are no images left to randomly select for testing; what remains are the last 250 images that have not yet been used for testing. In the end there should be tables of counting TP/FN/FP results from the 10 iterations of the counting process.

K-Fold Cross-validation of Disproportionate Number of Classes

The issues arise for multiclass classification especially when the set of GT images has a disproportionate number of classes and class combinations (e.g., 99000 images with T classes, 500 images with C classes, and 500 images with T&C [TC] classes). Stratification sampling has been introduced as a method to account for these disproportions and does suggest that appropriate sampling is required to tune models[4]. Further, it is important to eliminate selection bias when training deep learning classifiers[5]. However, sources do not explicitly discuss the sampling methods for deep learning that best account for multiclass classifiers which may have tendencies to specific errors for samples with specific class combinations. Therefore, it is important to eliminate selection bias by segregating the images with class-combinations in distinct sets given the potential tendency for a classifier to more likely create false decisions for pictures which include combinations of classes.

Continuing with this example, assume that there are 99000 T class images, 500 C class images, and 500 TC class images uniformly distributed in a 100000 sample set. If the number of C and CT class images in a training set fold are unfortunately disproportionality selected from the 100000 sample set, the variance of performance numbers increases, misrepresenting performance. To illustrate such sampling effects by exercising the limits of possibility with this example, a training set could have none of the C class appearances, while the test set has all the C class appearances. In this example, precision and recall would equal 0, misrepresenting performance. Other disproportionate ratios result in disproportionate performance metrics. This previous example illustrates the issues if stratification is not incorporated, notwithstanding the effect of not accounting for multiclass biases which may exist. This latter effect is accommodated if we take the following steps:

1. Create X bins, X = 2^(L-1) where L is the number of classes including the null class (i.e., E=empty class).
2. Segregate GT images by type of class and multiclass combinations of appearance possible in any image, into the appropriate X bin.

3. Apply stratified K-fold training/test set selection where each of the X bins is considered its own class for the sake of stratified K-fold validation. However, TP/FN/FP scoring shall only apply to each of the L classes.

Therefore, in the previous example, if we are applying this to our L=5 classes (T,S,F,C,E), X would equal 2^(5-1) distinct sets to be used for each of the K-fold Cross-validation test and training sets. This would result in 16 training sets and 16 test sets for each of 10 folds, for a total of 320 sample sets. We now ensure that every one of the X*K training sets will *always* contain 90% of the total GT appearances of each of the 5 given classes *as well as all 11 possible class combinations*. We also ensure that 10% of each of the 5 given classes *as well as all 11 class combinations always* appears in every one of the K test sets. This ensures a fair performance metric assessment of the classifier's ability to classify each of the identifiable classes, including effects of combined appearances of the class in an image.

Summary

This paper describes a method for validating performance of a multiclass classifier whose GT sets have disproportionate representations of desired classes and class combinations. We have shown, theoretically, that it is possible to have performance metrics equal 0 if the disproportionate representations of the desired classes and class combinations are not appropriately accounted in each of the K training and K test sets. Accommodating such an environment, including the disparity of GT samples which have class combinations, is not well publicized. It is critical to isolate the images with class-combinations in distinct sets, given the distinct likelihood for a classifier to bias false decisions for images which include combinations of classes. Now, given this publication, it should be readily available to all in the utility industry how to validate any such classifier accurately.

End of text

BIBLIOGRAPHY

[1] Hands on Machine Learning with Scikit-Learn and Tensorflow, Aurelien Geron, O'Reilly Media, pg 250

[2] R: Unleash Machine Learning Techniques, Raghav Bali, Dipanjan Sarkar, Brett Lantz, Cory Lesmeister, Packt Publishing, p 659

[3] The Elements of Statistical Learning, Trevor Hastie, Robert Tibshirani, Jerome Friedman, Springer-Hill, pp 241-245.

[4] Applied Predictive Modelling, Max Kuhn, Kjell Johnson, Springer-Hill, p 428

[5] Deep Learning, Josh Patterson and Adam Gibson, O'Reilly Media, pg 22