Efficient Imbalanced Multimedia Concept Retrieval by Deep Learning on Spark Clusters

Yilin Yan[1], Min Chen[2], Muhammad Saad Sadiq[1], Mei-Ling Shyu[1,*]

[1]Department of Electrical and Computer Engineering

University of Miami, Coral Gables, FL 33146, USA

[2]Computing and Software Systems

University of Washington Bothell, Bothell, WA 98011, USA

Emails: y.yan4@umiami.edu, minchen2@uw.edu, saadsadiq@miami.edu,

shyu@miami.edu

ABSTRACT

The classification of imbalanced datasets has recently attracted significant attention due to its implications in several real-world use cases. In such scenarios, the datasets have skewed class distributions while very few data instances are associated with certain classes. The classifiers developed on such datasets tend to favor the majority classes and are biased against the minority class. Despite extensive research interests, imbalanced data classification still remains a challenge in data mining research, especially for multimedia data. Our attempt to overcome this hurdle is to develop a convolutional neural network (CNN) based deep learning solution integrated with a bootstrapping technique. Considering the fact that convolutional neural networks are very computationally expensive coupled with big training datasets, we propose to extract features from pre-trained convolutional neural network models and feed those features to another full connected neutral network. Spark implementation shows promising performance of our model in handling big datasets with respect to feasibility and scalability.

Keywords: classification, deep learning, imbalanced data, semantic indexing, convolutional neural network (CNN), Apache Spark

Introduction

Skewness in data classes poses a significant challenge in major research problems pertaining to data mining and machine learning (Chen & Shyu, 2013; Chen & Shyu, 2011; Lin, Ravitz, Shyu, & Chen, 2007). Classes are rated as skewed or imbalanced when their data instances are non-uniformly associated to the class label. In real world cases, most applications have some degree of skewness inherently present in the data. Such datasets are often grouped into major and minor classes, where major classes have significantly greater numbers of instances associated with them as compared to minor classes. Some prominent imbalanced dataset use cases include fraud detection, network intrusion identification, uncommon disease diagnostics, critical equipment failure, and multimedia concept sensing. A number of famous classification methods are built to utilize the dataset statistics, which ends up being biased towards the majority classes. When identifying the minor classes, these classifiers often perform inaccurately even for very large datasets with considerable numbers of training instances.

Some notable frameworks aiming to solve this challenge are proposed in (Shyu, Haruechaiyasak, & Chen, 2003; Lin, Chen, Shyu, & Chen, 2011; Meng, Liu, Shyu, Yan, & Shu, 2014; Shyu, et al., 2003; Liu, Yan, Shyu, Zhao, & Chen, 2015; Yan, Chen, Shyu, & Chen, 2015). The authors of these frameworks, along with others, target this issue from two different perspectives. The first type is algorithm-based approaches where the authors propose new frameworks or improve the existing methods using both supervised and unsupervised techniques. The second, very different type is towards the manipulation of the data itself to reduce the skewness in the class attribution. However, the problem of imbalanced classes is far from being conquered, especially in multimedia data. Multimedia data is particularly difficult because of the various data types that are layered with spatio-temporal features.

One path to handle this challenging situation would be to employ solutions from other domains of machine learning such as deep learning. Deep learning is the name of a whole family of algorithms that use graphs with multiple layers of linear and non-linear transformations to develop hierarchical learning models (Wan et al., 2014). Several frameworks have been proposed using the deep learning techniques that show promising results in application domains such as automatic speech recognition (Swietojanski, Ghoshal, & Renals, 2014), computer vision (Chen, Xiang, Liu, & Pan, 2014), and natural language processing (Mao, Dong, Huang, & Zhan, 2014). However, deep learning methods have not been used to address the problems of class-imbalance. As illustrated in Section IV of our empirical study and also presented in (Sun et al., 2013; Snoekyz et al., 2013) on the TRECVID 2015 datasets, even the famous deep learning methods such as convolutional neural network (CNN) which outperforms a multitude of conventional machine learning techniques face difficulties when dealing with the class-imbalance problems. Moreover, for big datasets in multimedia data mining, deep learning methods are very expensive on computations. The method proposed in (Karpathy et al., 2014) took more than 30 days to train with 1755 videos. The authors were only able to successfully train the deep learning framework using a near-duplicate algorithm.

Toward such demands, our method is proposed to improve the TRECVID dataset confidence scores by a CNN based deep learning framework. In addition, a big data deep learning approach coupled with a bootstrapping sampling technique is proposed to create a balanced set of batches using the training dataset. To the best of our knowledge, bootstrapping has not been used with the deep learning frameworks. To further facilitate the capability of handling the class imbalance problem in big datasets, a distributed computation framework using Apache Spark is also implemented to bind the novel qualities of CNN with the bootstrapping procedures. The proposed framework has shown to be highly impressive and comparatively economical in classifying highly skewed multimedia datasets. The Spark-based distributed computing capability enables a scalable architecture that can mine unstructured key-value confidence scores of multimedia data.

The remaining of the article is organized in the following manner. The following section discusses the related work in skewed data classification methods, followed by some recent progresses in deep learning. Our proposed framework is introduced in the Framework section with its performance evaluated using the experimental results in the next section. The last section concludes the findings and develops the direction for future research.

Related work on classification for class-imbalance datasets

As mentioned in the previous section, class-imbalanced data classification methods can be sorted into two categories, namely the algorithm techniques and the data manipulation techniques. The first type of approaches, i.e., the algorithm based techniques, either propose to build new or improve existing algorithms to attain superior classification of imbalanced datasets. Consider the cost-sensitive learning methods as an example, where the method maximizes the cost functions of the data to increase the accuracy of class prediction. The intention behind these frameworks is that practical applications do not treat misclassified instances equally. These methods typically evaluate a cost matrix and utilize it in training the model. A relevant approach to cost-sensitive training is to modify the bias to give an advantage to the minority class (Unsworth & Coghill, 2006). Some frameworks using this approach do show the likelihood of improving the classification accuracy, but they are restricted to few application domains.

The second type of approaches that directly manipulate the data also have some notable frameworks. The methods of oversampling and downsampling the data are noteworthy to mention here (Yan, Liu, Shyu, & Chen, 2014). There are numerous implementations of these sampling based frameworks where the authors argue against whether oversampling has a better outcome than downsampling (Batista, Prati, & Monard, 2004). Oversampling methods reproduce duplicate or near duplicate positive data instances to balance out the dataset. Zhang et al. proposed a notable oversampling technique (Zhang & Wang, 2011; Chawla, Bowyer, Hall, & Kegelmeyer, 2002) but this method is potentially prone to overfitting. Alternatively, downsampling chooses from the pool of negative data instances to develop a model with a similar number of positive instances. There is an obvious advantage when only a subset of the majority class is used, but this also results in the loss of information because several critical instances may be ignored. Liu et al. came up with two methods to remove this disadvantage (Liu, Wu, & Zhou, 2009). The first one is called "Easy Ensemble" that samples various subsets within the majority class and the model is set to learn from each of these subsets. The output classification is an ensemble of these intermediate models. The second method is the "Balance Cascade" technique where the model learns in sequences. In each sequence, the accurately classified samples are removed from the next sequence.

Recent work on deep learning

With famous successful stories such as the Google AlphaGo platform (AlphaGo website), deep learning networks have proven to have a significant impact in a variety of research domains. It was originally evolved from the concept of artificial neural networks (ANNs), but is capable of achieving much higher performance metrics than some existing competing machine learning methods. Several deep learning methods have sprouted from the initial ANN concepts such as the deep belief network (DBN), restricted Boltzman machine (RBM), deep neural network (DNN), and many others (Wan et al., 2014). One of the promising off-shoots of the Deep Neural Networks approaches is called the convolutional neural network (CNN) (Swietojanski, Ghoshal, & Renals, 2014; Chen, Xiang, Liu, & Pan, 2014; Mao, Dong, Huang, & Zhan, 2014; Ji, Xu, Yang, & Yu, 2013; Jin, Fu & Zhang, 2014). CNN is a discriminatory deep learning architecture that has gained much popularity in computer vision and object recognition. CNN is composed of modules and each module is mainly made up of two layers, namely the convolutional layer and the pooling layer. The convolutional layer shares many weights and passes its output to the pooling layer, and the pooling layer then subsamples the output and effectively cuts down the data rate. Although CNNs are the preferred choice in various domains, they have yet to challenge the class-imbalance problem. Therefore, in this paper, we implement CNNs on a highly class imbalanced dataset and strive to improve the performance over the conventional state-of-the-art methods.

Since AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) won the top-5 test error rate in the ILSVRC 2012 contest on ImageNet, CNN is proven to suppress the traditional ensemble framework with classical features like sparse SIFT and pyramid pooling. Correspondingly, a deep and large CNN was trained to classify more than one million high-resolution images in the ImageNet LSVRC-2010 competition into a thousand different classes, and a variant of the well trained model was entered to the ILSVRC-2012 contest. Two years later, the width and depth of CNN were increased while keeping the computational budget constant by carefully crafted designs in Caffe (Jia et al., 2014). The quality of the 22 layers deep network, GoogLeNet (Szegedy et al., 2014), was assessed in the competition of detection and classification. With CNN becoming more of a commodity in the computer vision field, many research groups attempted to improve the original architecture of AlexNet. Simonyan et al. (Simonyana & Zisserman, 2014) from Oxford’s renowned Visual Geometry Group (VGG) also got good results in ILSVRC 2014 by increasing the CNN depth with very small convolution filters and showed a significant improvement by pushing the depth to 16–19 weight layers. Yet, the overall trend of CNN is making the networks deeper and deeper. In a popular recent work, ResNet (He, Zhang, Ren, & Sun, 2015), the neural network depth is eight times larger than VGG. With the proposed residual learning framework and substantially deeper network, they won the 1st place on the ILSVRC 2015 classification task. Although CNN with thousands of layers have been developed recently, very few of them focus on the imbalanced datasets.

With the rapidly growing of neural network depths and modern multimedia datasets, more computational power and efficient clusters are needed. Several data processing engines are turning towards generalized MapReduce frameworks (Dean & Ghemawat, 2008). Apache Spark (Zaharia et al., 2012) was able to develop an open-source distributed processing engine called Spark that increases the capability of conventional MapReduce use-cases. This extension has contributed to two common classes, the iterative algorithms (i.e., machine learning graphs) and the interactive data mining multiple computation chains. Spark naturally dovetails into Hadoop ecosystems with the HDFS storage and Yarn or Mesos as the managers. Yahoo developed a spark package that brings deep learning to Hadoop and Spark clusters named CaffeOnSpark (CaffeOnSpark website) which supports neural network model training, testing, and feature extraction as a distributed extension of Caffe. Though a small number of this kind of tools were proposed in the past two years, most of them are still in the development stage with unstable functions and cannot be directly applied to the current CNN frameworks.

Convolutional neural network (CNN) structures

As to be introduced in this section, the deep learning frameworks like CNN have been proven to be one of the most significant developments recently and is famous for its ability to create multiple levels of training and abstraction that help to understand the data easily. This section discusses how the CNN framework can be utilized for the multimedia class-imbalanced datasets.

CNN is a subdivision of the deep neural network chain in deep learning that, at the root, are variants of multilayer perceptron. CNNs are configured to utilize minimum resources in preprocessing (Hastie, 2005; LeCun, Bottou, Bengio, & Haffner, 1998). This is done with two techniques: the first is to limit the links among the invisible sections and the input section so that each invisible section links to only a subset of the input called feature maps. The motivation behind the technique of having locally linked networks is taken from the visual cortex where neurons also have local receptive fields (Kanel, 2009). The second technique is to develop simplified computations of images. Since natural images have the tendency of being stationary, the statistics of the different regions of natural images are similar. The second technique takes the advantage of this, utilizes a random subset of trained features, and convolves them to acquire feature activations of the remaining image. Then these acquired features can be used either directly or as ensemble statistics for classifying the data. The ensemble statistics have the characteristics of being comparably very low dimensionality and not overfitting the model as well.

Generally speaking, a CNN model consists of three kinds of layers, which are convolutional, pooling, and fully connected layers (Bouvrie, 2006). The convolutional layer is composed using several feature maps as defined in Equation (1). The feature map of the *l*th layer and *j*th feature batch X_j^l is evaluated by convolving the feature maps of its preceding layer X_j^{l-1}. The convolution process uses the activation function *f* with trained kernels K_{ij}^l and an additive bias b_j^l. The first layer X_j^1 corresponds to the input data, a logistic function is selected as the activation function *f* that corresponds an assortment of the input maps.

$$X_j^l = f\left(\sum_{i \in M_j} X_i^{l-1} * K_{ij}^l + b_j^l\right), l \geq 2; \qquad (1)$$

Here, a feature map is divided into several batches M_j, where M_j represents the data batches and *i* is the index of each those batches. The pooling layer takes in the input features as given in Equation (2) and outputs a subsampled version of it. Here, the operation “pool” stands for a pooling procedure that evaluates the ensemble statistics of the input maps, β_j^l depicts the multiplicative bias, and b_j^l shows the additive bias. The pooling layer is normally placed after each convolutional layer and it typically is designated as a mean or max pooling procedure.

$$X_j^l = f\left(\beta_j^l pool(X_j^{l-1}) + b_j^l\right), l \geq 2. \qquad (2)$$

The fully-connected layer is developed to be the high-level reasoning layer in the network. It is placed after or close to the terminal layers of the neural network. All neurons from the earlier layers are then connected to each neuron in the fully-connected layer.

Proposed imbalanced multimedia concept retrieval framework: CNN with bootstrapping

Our deep learning framework is contrasting from the negative bootstrap framework developed in (He, Zhang, Ren, & Sun, 2014) that joins random sampling and adaptive selection to recursively search the related negatives. The proposed bootstrapping sampling technique integrates oversampling with decision fusion to improve the CNN’s classification accuracy on multimedia data that may or may not have skewed class attributions.

Even after the tremendous success of deep learning frameworks, to the best of our knowledge, only a handful of papers target the challenging problem of skewed class multimedia data. As a matter of fact, directly applying deep learning methods on imbalanced data ends in a very bad classification accuracy. This is illustrated by the empirical study results comparing the balanced and imbalanced datasets in Figure 1 where x-axis is for the number of training iterations while y-axis is for error rate, and the error rate of the prediction is compared to the increasing number of iterations of a CNN deep learning network. In the case of balanced datasets in Figures 1(a) and 1(b), the error rates steadily decrease to definite points. However, when the CNN is used with imbalanced datasets as shown in Figures 1(c) and 1(d), the prediction error rates waver about plateau points. The reason behind this oscillating error rate is because the deep learning training stages allot the training data into groups. The grouping becomes unfair when the data is imbalanced and some groups may end up containing all negative data instances and no positive data instances. The result of this imbalanced class distribution is a low accuracy classification model.

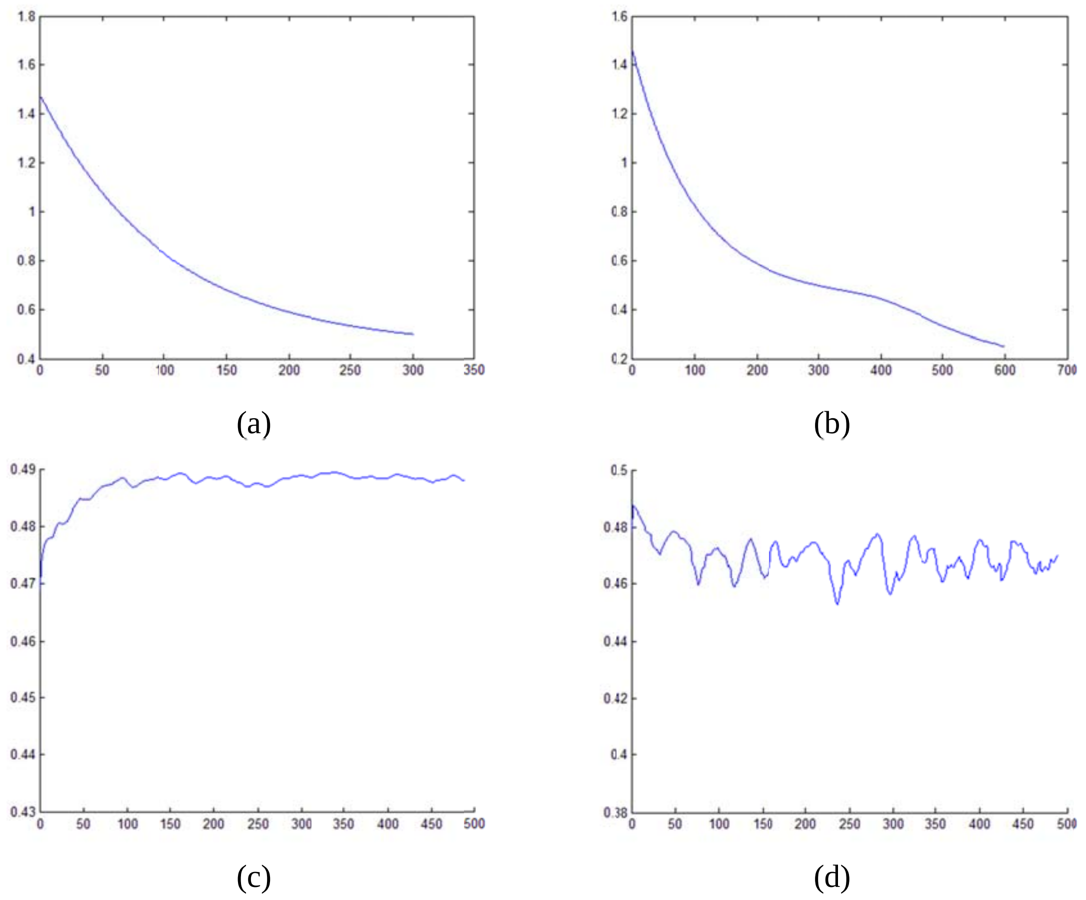


Figure 1. [Difference in the total error rate produced from (a) & (b) balanced datasets and (c) & (d) imbalanced datasets]

Conventional bootstrapped samples have the imbalanced dataset with $n \gg m$, where n and m are the numbers of negative and positive data instances, respectively. Our proposed framework generates batches of s_{neg} and s_{pos} data instances to balance out the ratio $r = s_{neg}/s_{pos}$, where s_{neg} is the number of negative data instances and s_{pos} the number of positive ones. Therefore, totally M batches will be generated, where:

$$M = \left\lfloor \frac{n}{s_{neg}} \right\rfloor \quad (3)$$

Another way to see it is when n is not exactly divisible by s_{neg}, any remaining negative data instances will be removed in the training stage. Since the total count of negative data instances n in the training set is large and the batch size $s = s_{neg} + s_{pos}$ is typically small (i.e., resulting in a small s_{neg}), the removed negative data instances become negligible comparative to the data instances used in the training stage. Then from the m positive data instances, one positive data instance is randomly chosen s_{pos} times and combined with s_{neg} negative data instances for each batch. This process is repeated I times to produce the batches in each learning repetition until the error rate converges. This random stochastic process assures an equivalent probability for each positive data instance to be selected and trained with various negative data instances and eventually avoid overfitting. Table 1 illustrates the discussed process. In each repetition process, the bootstrapping process produces a pseudo balanced training set from the original imbalanced dataset, which can be then used by the deep learning model for learning.

Table 1. Proposed module of CNN with bootstrapping.

PSEUDO CODE OF CNN WITH BOOTSTRAPPING

```
1.  Split the training set into a positive set pos and a negative set neg
2.  Divide neg into $M$ batches, each with $s_{neg}$ negative data instances
3.  for 1:$I$
4.    for 1:$M$
5.        for 1:$s_{pos}$
6.            randomly pick one data instance from pos;
7.        end for;
```

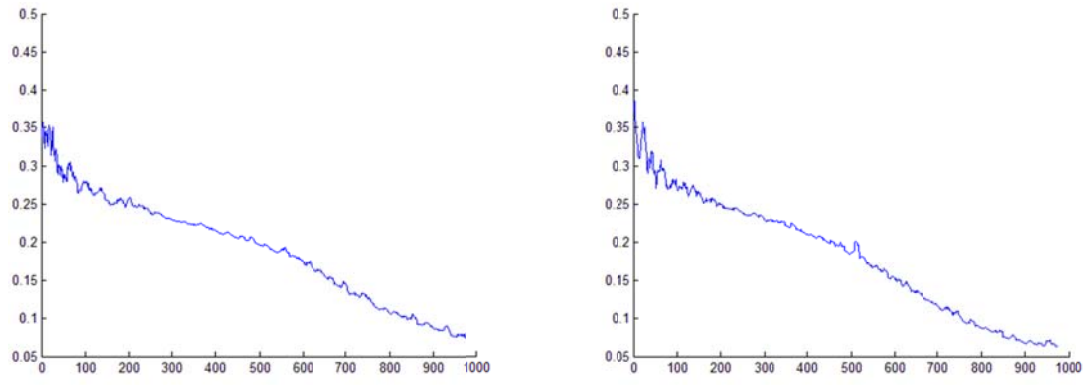
```
8.        combine the data instances in pos and neg together;
9.     end for;
10.    Train a CNN model;
11. end for;
12. end;
```

Let the size of each input be $m \times m$ such that a four mid-layer CNN forms as depicted in Table 2, where k_L represents the number of mask neurons applicable on a given subgroup of input values and $n_L \times n_L$ indicates the size of each mask in the L^{th} convolution layer. The output from the L^{th} convolution layer is given to the L^{th} pooling layer and it is split into a group of non-overlapping rectangles of size $p_L \times p_L$, where the pooling operations are applied for downsampling. Furthermore, the bootstrapping method explained earlier is then used to generate *N* batches of balanced training instances that are given to the first layer of CNN in iterations. The input layer is followed by two convolutional layers, and then followed by their respective mean pooling layers. The first convolutional layer produces the inner product of the $k_1 \times (n_1 \times n_1)$ masks and passes the output to the first mean pooling layer. Mean pooling layers summarize the outputs of the neighboring subsets of masks in the same kernel map. The output of the pooling layer is passed as the input to the second convolutional layer. This is followed by the mean pooling layer using the same process as mentioned earlier but with different mask sizes. The size of the vector of the final CNN layer denotes the number of classes attributed to the data. Our experiment performs binary classification, and thus the size is set to 2.

Table 2. Training parameters for CNN.

Layer	Layer size	Output size
Input (m*m)		
Convolution 1	k_1*n_1*n_1	k_1*(m-n_1+1)*(m-n_1+1)
Pooling 1	p_1*p_1	k_1*(m-n_1+1)/p_1*(m-n_1+1)/p_1 [let m_2=(m-n_1+1)/p_1]
Convolution 2	k_2*n_2*n_2	k_2*(m_2-n_2+1)*(m_2-n_2+1)
Pooling 2	p_2*p_2	k_2*(m_2-n_2+1)/p_2*(m_2-n_2+1)/p_2
Output		2

Finally, the consequent convolution masks and weights are used for classification and the class receiving the highest score will be attributed to the test data instance. The benefit of the bootstrapping method can be observed from the prediction error rates shown in Figure 2 as it is applied to the imbalanced dataset of Figures 1(c) and 1(d), respectively. The descending error rates illustrate the effectiveness in the convergent process.



(a) (b)

Figure 2. [Total error rates convergence generated from imbalanced datasets using our bootstrapping method]

Integration of CNN and low-level features

The training time of CNN is notorious for being computationally taxing. For example, it took one month to train 1755 videos to achieve sufficient performance metrics (Karpathy et al., 2014). It is observed in the literature that training a deep learning method is substantially longer if provided with the raw data at the input layer. To improve the efficiency of CNN, we first propose a different approach by using the low-level multimedia features that are much smaller than the raw signal data. This key process here is that we propose to feed the low-level features into the CNN directly to substantially reduce the *m* value (18 in the proposed experiment) and greatly improve the processing times. These low-level features are composed of Haar (Verma & Maru, 2009), HOG (Yan et al., 2014), HSV, YCbCr (Sural, Qian, & Pramanik, 2002), and CEDD (Chatzichristofis & Boutalis, 2008). These are chained into a feature vector which is then transformed to a matrix using PCA (Principle Component Analysis). This transformation is required because CNN does not support one-dimensional vectors and the features are fed as an 18 by 18 matrix. The sizes of the matrix and masks in Table 3 are decided based on empirical studies and could be adjusted with different feature dimensions and datasets.

The internal deep learning process of the CNN is similar to what is described in the previous section, except for the pooling layers that are removed because the low-level features are not necessarily stationary in every iteration. Table 3 illustrates the detailed training parameters used in the proposed framework. Since we have earlier reduced the dimensions of the input features, a relatively small mask size can be applied, in comparison to that in (CaffeOnSpark website).

Table 3. Training parameters.

Layer	Mask size	Output Size
Input (18*18)		
Convolution 1	6*3*3	6*16*16
Convolution 2	9*3*3	9*14*14
Output		2

Deployment of the proposed framework on a Spark cluster

A shortcoming of using low-level features is potentially losing information. Thus, another idea of making an efficient framework is to build it on top of a big data processing system. In this paper, we built our model on top of our dedicated Spark cluster. The cluster is running most recent versions of the required distributed big data infrastructure, i.e., Apache Hadoop 2.7.3 with Yarn and Apache Spark 2.0. The developed Spark cluster serves as the primary test bed cluster for deep learning and distributed processing experiments (Yan, Shyu, & Zhu, 2016; Yan, Zhu, Shyu, & Chen, 2016; Yan et al., 2016).

Figure 3 illustrates the core infrastructure of the cluster. There are 4 nodes in total with our master node connected to a dedicated class-c dedicated IP. The overall cluster configuration is heterogeneous but is normalized to the least performing node in the infrastructure. Each node is setup to instantiate 2 workers with 4 GB of memory and 1 TB of storage. The data files were replicated across the three Hadoop HDFS instances in all data nodes, namely data node 1, data node 2, and data node 3 as shown in Figure 3.

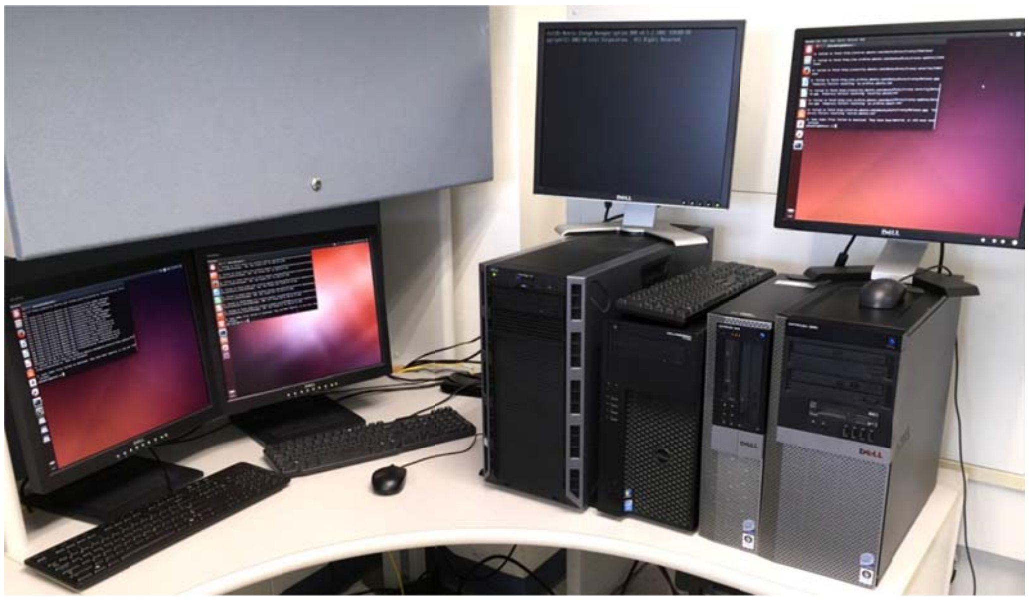


Figure 3. [The infrastructure of our spark cluster]

The main abstraction that Spark provides is a resilient distributed dataset (RDD), an in-memory storage abstraction of frame confidence score elements partitioned across the nodes of the cluster that can be operated on in parallel. These RDDs were created from confidence score files present in Hadoop Distributed File System (HDFS). The proposed CNN framework ran on Spark as tasks coordinated by the master node in the cluster which has a driver program. Linear speedup was achieved by modifying the Spark configuration and setting parallelism to the number of cores present in the cluster to utilize the cluster to its maximum capacity. It is recommended that a maximum of 2 to 3 tasks are to be run on a CPU core at an instance of time. The number of tasks executed in parallel on a node is equal to the number of cores in the corresponding node. Spark automatically sets the number of map tasks to run on each file according to the number of partitions present. A partition is defined by each file which is loaded from the HDFS. Executors were started on each node of the cluster to perform the tasks.

Deep learning methods are notorious for computationally expensive and impractical for streaming data. Our attempt to overcome this challenge is to use a distributed environment and Spark. By the empirical testing and evaluation, it was observed that the same neural network implementation using Java in Apache Spark 2.0 achieved 400% speed improvement over Matlab 10 performed on the same cluster.

There are a lot of use-cases where multi-core or GPU based processing or conventional HPC systems may be significantly faster than any Spark implementation. We have to take all the feasibility cases into account and argue our case of building a system to make positive forward progress in our research. Since Spark is only good with recursive statements and streaming inputs, in the case of classifying data, MapReduce will probably do a competitive job to Spark due to the fact that there is only one time read involved from the hard disk.

Although the Spark clusters are proven to be suitable for recursive computing, how to distribute the neural network training remains as a challenging issue. Since all parameters in a certain layer are updated after training each mini batch, most popular neural network models cannot be deployed on distributed computing clusters to run parallel processing. Another problem to deploy the neural network frameworks on Spark is that Spark supports Scala and Java stably, but only partially supports

Python. Hence, we use a deep learning tool written in java called Deeplearning4j to implement parallel computing on training the neural network models.

The basic idea of training a neural network model on Spark is parameter averaging. As drawn in Figure 4, the input data is divided into several subsets based on the configuration of the master node. With the same as the regular neural network training steps (please note that the "data split" here is different from the "batch"), the Spark driver (i.e., the master node) starts with a randomly generated parameter set and an initial network configuration. For each subset of the training data, the master node distributes the parameters, configuration, and network updater states to all slave nodes. Then, each slave node trains its own portion of the subset and updates the parameters as well. After several iterations, the parameters and states are sent back to the master node which will average the parameters and states to update the trained neutral network. Then, the average values are distributed to the slave nodes as well as the workers again for further training.

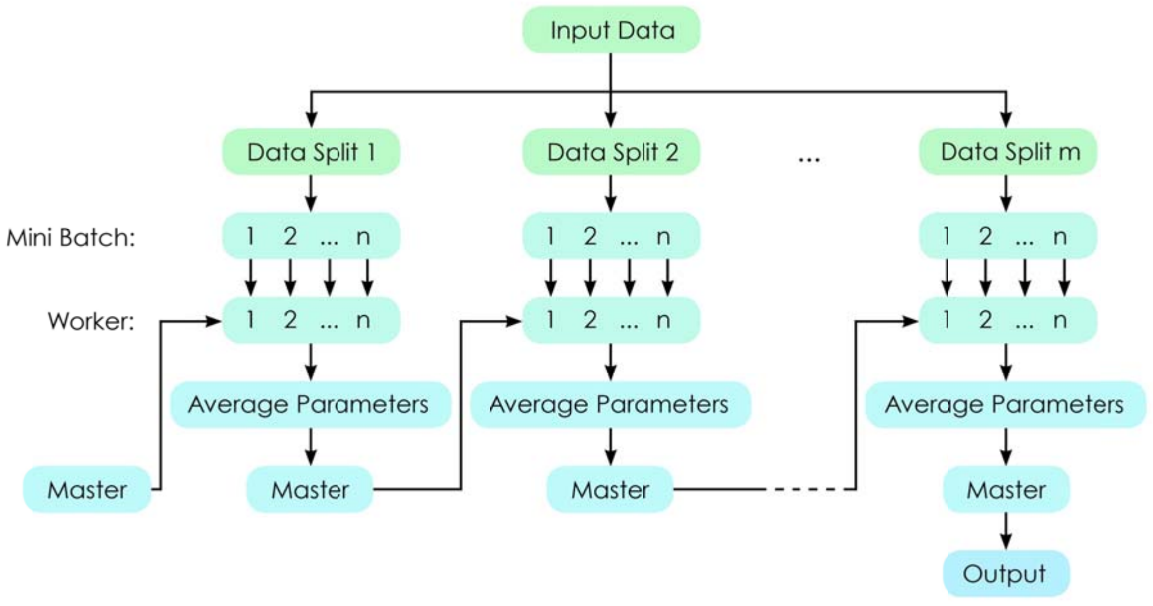


Figure 4. [The flowchart of training CNNs on our Spark cluster]

In this study, we identify the best available features to identify the skewed and imbalanced classes and improve the classification of the imbalanced datasets. It is an industry best practice to start with the current best features and further improve the performance by processing the skewed and imbalanced classes. Recently, those features extracted from pre-trained deep learning models are proven to outperform traditional low-level features. In this paper, we use a pre-trained and fine-tuned CNN model on the ImageNet data, Alexnet (Krizhevsky, Sutskever, & Hinton, 2012), for keyframe feature extraction. The Alexnet structure is well trained and proven with great performance. It contains five convolutional layers and three fully-connected layers and our CNN features are extracted from all the training and testing keyframes from the output layer, i.e., the 8th layer with one-thousand dimensions. These features are finally fed to a neural network with two fully connection layers, where the first layer contains 100 neurons and the second one is composed of 10 neurons.

Performance measurement

In general, a classifier is evaluated by a confusion matrix as illustrated in Table 4. The columns are the predicted class and the rows are the state of nature (actual class). In the confusion matrix, TP (True Positives) and FP (False Positives) represent the numbers of positive data instances that are correctly or incorrectly classified, respectively. Similarly, TN (True Negatives) and FN (False Negatives) indicate the numbers of negative data instances being correctly or incorrectly classified, respectively. For performance comparison, the precision and recall metrics (Buckland & Gey, 1999) are commonly used and are derived from the confusion matrix in Table 4 and Equations (4) and (5).

Table 4. Confusion Matrix.

	Predicted Positive	Predicted Negative
State of nature Positive	True Positives (TP)	False Negatives (FN)
State of nature	False Positives	True Negatives

Negative	(FP)	(TN)

The recall and precision goals can often be conflicting, since the increase of true positive data instances for the minority class may also increase the number of false positive data instances, which will reduce the precision. For imbalanced data classification, the recall value is normally considered a more important criterion because it is more desirable to detect as many interesting events as possible, even at the expense of adding a reasonable number of false positive data instances. For example, users often want to locate all possible frauds in banking operations followed by a manual double check to root out false alarms, instead of missing true scams. In addition, *F-score*, also known as F_1 measurement or *F-value*, captures the trade-offs between precision and recall, and is considered an objective and ultimate quality metric of a classifier which is defined in Equation (6).

$$precision = \frac{TP}{TP+FP} \tag{4}$$

$$recall = \frac{TP}{TP+FN} \tag{5}$$

$$F-score = 2 \times \frac{precision \times recall}{precision + recall} \tag{6}$$

Evaluation on the UCF11 dataset

First, our proposed model is tested on a relatively balanced video dataset, UCF YouTube Action (UCF11) dataset (Liu, Yang, & Shah, 2009), to prove the efficiency of feeding the low-level features to the CNN models. UCF11 includes videos collected from YouTube with various problems like non-static background, low video quality, camera motions, poor illumination conditions, etc. It is a relatively balanced dataset as compared to the TRECVID dataset and contains 11 action categories: basketball shooting, biking/cycling, diving, golf swinging, horseback riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volley ball spiking, and walking with a dog. This data set is very challenging (Liu, Luo, & Shah, 2009) due to the large variations in camera motion, object appearance and pose, object scale, viewpoint, cluttered background, illumination conditions, and etc.

For each category, the videos are grouped into 25 groups with more than 4 action clips in it. The video clips in the same group share some common features, such as the same actor, similar background, and similar viewpoint. The STIP features (Laptev & Lindeberg, 2003) are extracted from each UCF11 video. STIP is built on the idea of the Harris and Forstner interest point operators to detect local structures in space-time where the image values have significant local variations in both space and time. STIP features can be obtained by estimating the spatio-temporal extents of the detected events and computing their scale-invariant spatio-temporal descriptors. The dimensions of the STIP features are reduced to 32 from 162 by applying the principle component analysis (PCA) technique for fast computation purposes. In the video representation part, 256 Gaussian components in GMMs (Gaussian Mixture Models) are used and the leave-one-out cross validation scheme is employed. Since UCF11 is a relatively balanced dataset, in the bootstrapping step, a small number of data instances are randomly picked from each category (5 in our experiment) to form the batches for CNN training. Figure 6 shows some example frames from the UCF11 dataset.

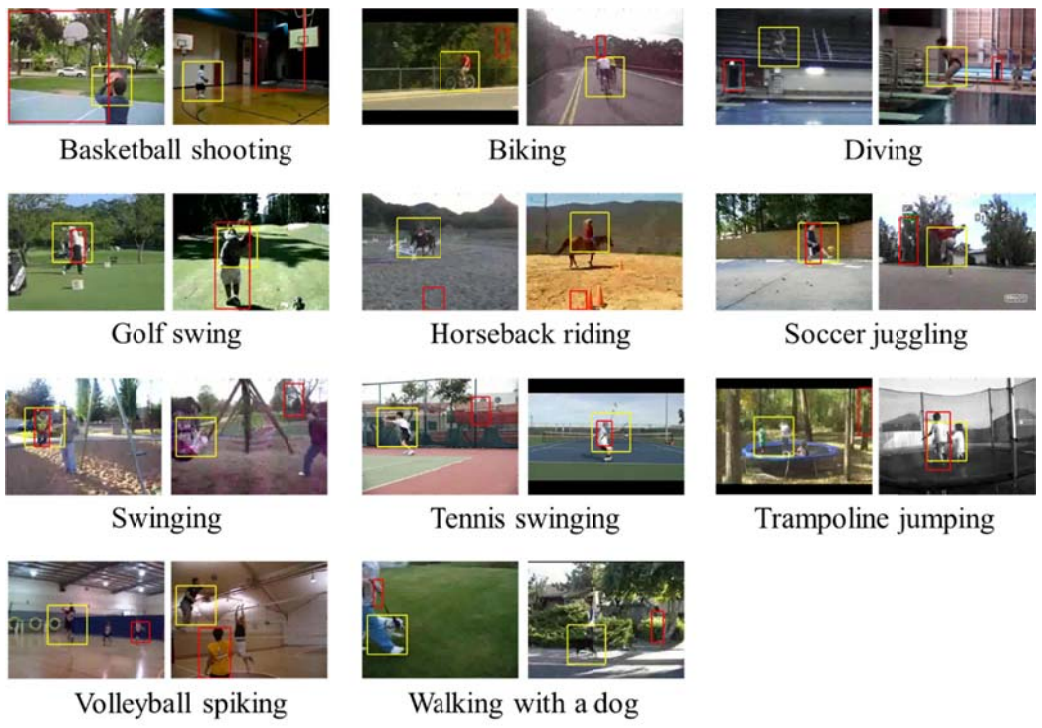


Figure 5. [Example of UCF11 (UCF YouTube action) data set with approximately 1,168 videos in 11

categories]

Table 5 shows the confusion matrix of applying our framework to the UCF11 dataset. Here, "Bas" denotes basketball shooting, "Bik" is for biking/cycling, and so on. The vertical labels are the ground truth, i.e., the actual labels; while the horizontal side shows the prediction labels. The number in each grid shows the percentage of the data instances. For instance, the number "85" shows that 85 percent of the 'horseback riding' testing instances are correctly identified; while the number "1" shows that 1 percent of the horseback riding data instances are misclassified as soccer juggling. Table 6 shows the performance comparison between our approach and three other state-of-the-art methods. Specifically, (Perez et al., 2012) used the combination of Histograms of Gradients into orientation tensors and applied SVM as the classifier. In (Liu, Luo, & Shah, 2009), motion features based on the ROI (Region of Interest) estimation and AdaBoost were used to integrate all the heterogeneous yet complementary features for recognition. In (Mota et al., 2014), SVM was applied to a tensor motion descriptor with optical flow for action recognition.

Table 5. Confusion Matrix of the UCF11 Dataset.

	Bas	Bik	Div	Gol	Hor	Soc	Swi	Ten	Tra	Vol	Wal
Basketball shooting	55	5	3	8	1	1	2	13	0	11	1
Biking	1	73	0	0	10	0	3	3	2	2	5
Diving	5	2	76	1	1	1	2	1	1	6	3
Golf Swing	12	1	1	82	0	1	2	2	0	0	0
Horse Riding	1	6	1	0	85	1	1	1	1	0	6
Soccer Juggling	4	1	1	4	5	63	6	5	1	4	5
Swinging	1	4	4	1	1	1	79	0	4	3	2
Tennis Swing	8	1	1	8	4	3	2	72	1	1	1

Trampoline Jumping	1	0	1	0	2	9	8	1	77	1	1
Volleyball Spiking	7	1	2	1	0	2	1	8	0	79	0
Walking with a dog	2	7	2	3	20	1	2	5	2	0	54

As shown in Table 6, our approach achieves the best accuracy rate among all the methods. This experiment clearly proves that while our framework aims to address the challenges caused by a highly imbalanced data distribution, it is also very effective in classifying relatively balanced datasets.

Table 6. Result Comparison for the UCF11 Dataset.

Group	Accuracy
Perez et al. (2012)	68.9%
Liu, Luo, & Shah (2009)	71.2%
Mota et al. (2014)	72.7%
Our Framework	**72.8%**

Experimental results on the TRECVID dataset

In order to demonstrate the effectiveness of our proposed framework for imbalanced multimedia data classification, the TRECVID dataset (Awad et al., 2016), a large-size benchmark dataset with highly skewed data distribution, is used in the experiment. In particular, the IACC.1 dataset from the TRECVID 2015 datasets (Over et al., 2015) is used. The semantic indexing (SIN) task in TRECVID 2015 aims to recognize the semantic concept contained within a video shot, which can be an essential technology for retrieval, categorization, and other video exploitations. Here, the concepts refer to the high-level semantic objects such as a car, road, and tree. Figure 6 shows four sample keyframes with the labeled concepts. There are several challenges such as data imbalance, scalability, and semantic gap. As a result, traditional deep learning approaches, including CNNs, often perform poorly on the TRECVID dataset due to the problem of under-fitting, huge diversity, and noisy and incomplete data annotation (Sun et al., 2013; Snoekyz et al., 2013). Please note that the data imbalance degrees of different concepts vary in the TRECVID dataset, and thus a fixed batch size may not be suitable for every testing concept. To address this issue, the batch size is chosen dynamically based on the number of positive data instances in the training set. In this experiment, the batch size is set to be twice bigger than the number of positive training data instances.

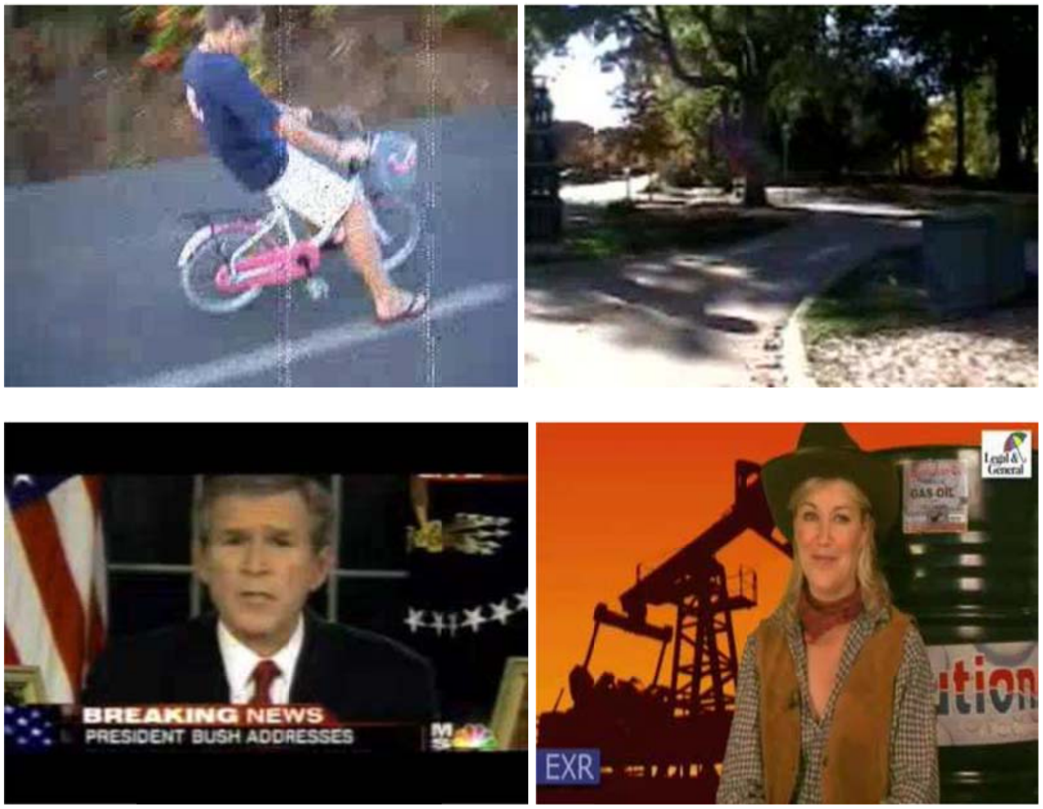


Figure 6. [Sample keyframes with annotated concepts in the TRECVID dataset: the concepts are bicycling, tree, politics, and face, respectively]

Here, the TRECVID dataset is chosen mainly because it contains a large number of data instances and is highly imbalanced (Smeaton et al., 2006). In the selected IACC.1 dataset, a total number of 262,911 data instances are used for training; while 113,046 data instances are used for testing. Our proposed framework is evaluated on 84 concepts with severely skewed data distributions and P/N ratios lower than five ten-thousandths, where the P/N ratio is the ratio between the number of positive data instances and the number of negative data instances. As indicated in (Qiong, Li, & Zhihua, 2009), in imbalanced data classification, the recall metric is considered more important than precision and the F-score represents the trade-off between precision and recall. As shown in Figures 7 and 8, the recall and F-score values of our proposed framework using the low-level features and the features from the pre-trained CNN models are compared with the scores from TiTech (Tokyo Institute of Technology) that achieved the best performance in the semantic indexing task several times in the past years (Inoue et al., 2011; Inoue & Shinoda, 2012).

In case of the TRECVID confidence score evaluation, we have to work with the unstructured key-value pairs of the TRECVID video shots. There is a need to store the massive TRECVID multimedia data, in the order of several Terabytes, with redundancy over the years since 2003 until recent. We have stored the TRECVID video frames as well as the extracted confidence scores that are continuously used in our models to compare with previous datasets or to train for the recent competition. The photos and video frames can be stored in the HDFS and processed using the Spark engine and the confidence scores can be assigned and stored back in the HDFS. The resultant confidence scores are unstructured key-value pairs that need to be stored in the HDFS based redundant data store and accessed for data mining processing. Therefore, Spark befits as a perfect candidate solution to our problem.

From the results drawn on Figures 7 and 8, our F-scores generated from the low-level features are higher than those of the TiTech group for two thirds of the 84 concepts; while the results by those features from the pre-trained CNN models perform better than four fifths of the TiTech scores. For the recall measurement, both of our frameworks generate better results for almost every concept. The only exception is when using the low-level features, they may fail to identify a true positive data instance due to the noisy data annotations and information lost in feature extraction. It is also worth noting that for 50 concepts, the TiTech group can only locate zero or one true positive data instance;

while our approach reaches more than 0.628 recall value on average. This clearly demonstrates the effectiveness of integrating CNNs with the bootstrapping strategy in our proposed framework for imbalanced multimedia data classification, especially on the fact that the study in (Batista, Prati, & Monard, 2004) showed that the performance of CNNs is far worse than all other classifiers the authors tried on the TRECVID dataset.

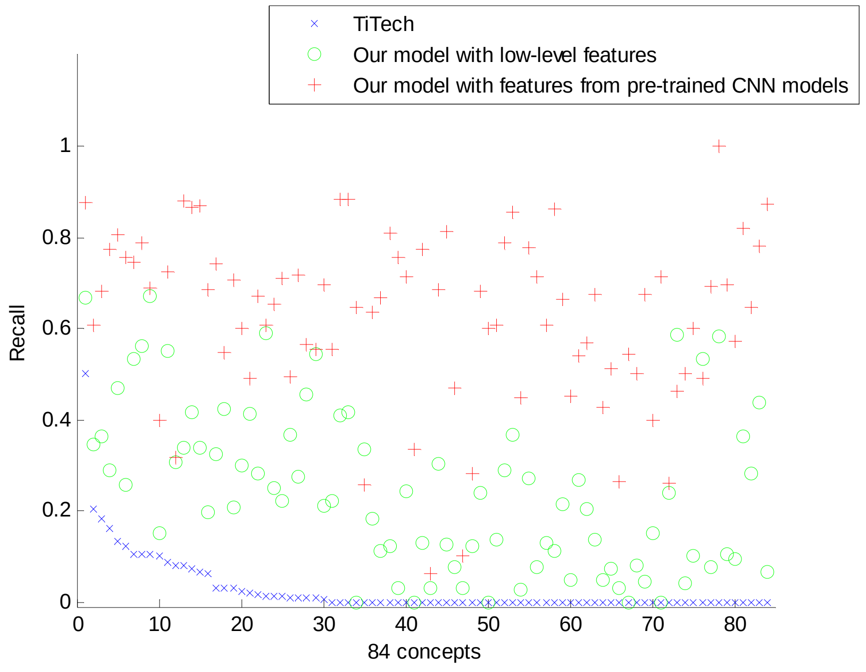


Figure 7. [Recall comparisons on all imbalanced concepts]

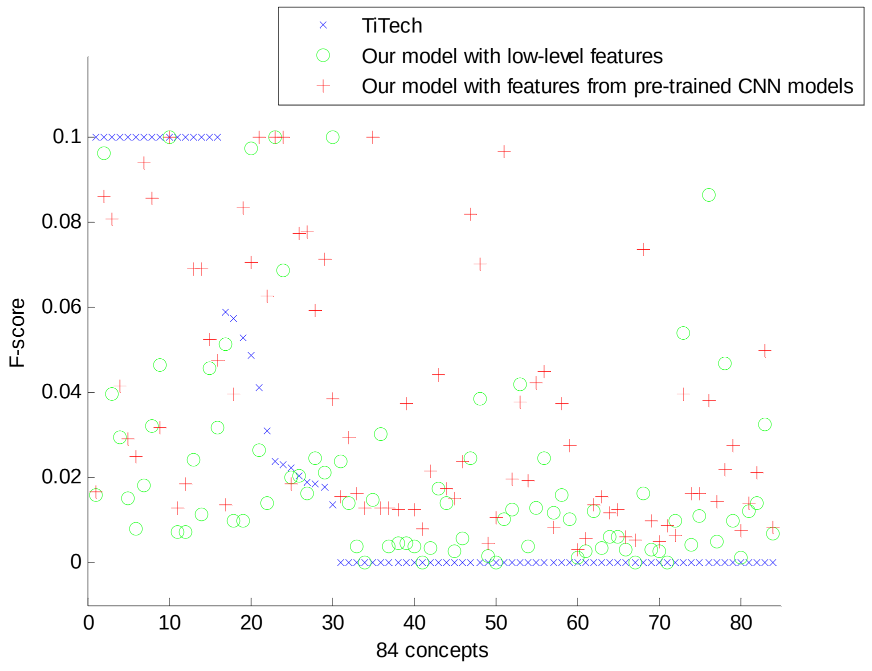


Figure 8. [F-score comparisons on all imbalanced concepts]

Conclusions and future work

In this paper, we proposed to extend the convolutional neural networks based deep learning technique by incorporating a bootstrapping algorithm. Moreover, to achieve faster computation speeds and better handling of unstructured key-value pairs of the TRECVID video data, we harnessed the power of Apache Spark. Our Spark system is implemented on a dedicated Spark cluster developed solely for the computational needs of our research group. In the bootstrapping stage, pseudo balanced training batches are rendered and inserted into the CNN for classification. The experimental results establish the effectiveness of the proposed framework for accurately classifying highly imbalanced multimedia data. Different from many existing methods in deep learning that take the required raw media data in the input layer, our deep learning framework works efficiently on the low-level features, which largely reduces the required training time in deep learning. Furthermore, a computational boost is achieved with the power of distributed computing using Apache Spark and better information retrieval results are generated by the features from the pre-trained CNN models.

Though we propose a powerful imbalanced big data processing system using Spark in this paper, running deep learning algorithms on GPU is much more efficient than on CPU. Therefore, it is better to extend the system for accelerating deep learning on Spark applications using GPUs. Since GPUs provide both high-computation capabilities and high-memory bandwidth, they can be used to accelerate both computation-intensive and memory-intensive Spark jobs. In the future, we plan to enhance our system and run deep learning applications on distributed GPUs with Spark.

REFERENCES

G. Awad, J. Fiscus, M. Michel, D. Joy, W. Kraaij, A. F. Smeaton, G. Qunot, M. Eskevich, R. Aly, and R. Ordelman. "Trecvid 2016: Evaluating Video Search, Video Event Detection, Localization, and Hyperlinking," In Proceedings of TRECVID 2016, NIST, USA, 2016.

G. E. Batista, R. C. Prati, and M. C. Monard, "A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data," SIGKDD Explorations, vol. 6, issue 1, pp. 20-29, June 2004.

J. Bouvrie, "Notes on Convolutional Neural Networks," Online technical report, 2006.

M. Buckland and F. Gey, "The Relationship Between Recall and Precision," Journal of the American Society for Information Science, vol. 45, issue 1, pp. 12-19, January 1999.

S. A. Chatzichristofis and Y. S. Boutalis, "CEDD: Color and Edge Directivity Descriptor: A Compact Descriptor for Image Indexing and Retrieval," In Proceedings of the 6th International Conference on Computer Vision Systems, pp. 312–322, Berlin, Heidelberg, 2008.

N.V. Chawla, K.W. Bowyer, L.O. Hall, and W.P. Kegelmeyer, "SMOTE: Synthetic Minority Overbootstrapping Technique," Journal of Artijcial Intelligence Research, vol. 16, pp. 321-357, 2002.

C. Chen and M.-L. Shyu, "Clustering-based Binary-class Classification for Imbalanced Data Sets," In Proceedings of the 12th IEEE International Conference on Information Reuse and Integration, pp. 384-389, Las Vegas, Nevada, USA, August 2011.

C. Chen and M.-L. Shyu, "Integration of Semantics Information and Clustering in Binary-class Classification for Handling Imbalanced Multimedia Data," Edited by Tansel Ozyer, Keivan Kianmehr, Mehmet Tan, and Jia Zeng, Information Reuse and Integration in Academia and Industry, Chapter 14, Springer Verlag, 2013.

X. Chen, S. Xiang, C.-L. Liu, and C.-H. Pan, “Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks,” IEEE Geoscience and Remote Sensing Letters, vol.11, no.10, pp. 1797-1801, Octorber 2014.

D. Jeffrey and S. Ghemawat. “MapReduce: Simplified Data Processing on Large Clusters,” Communications of the ACM, vol. 51, issue 1, pp. 107-113, 2008.

T. Hastie, “Neural Networks,” Edited by P. Armitage and T. Colton, Encyclopedia of Biostatistics, John Wiley & Sons, 2005.

K. He, X. Zhang, S. Ren, and J. Sun, “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition,” In Proceeding of the European Conference on Computer Vision, pp. 346-361, Zurich, Switzerland, September 6-12, 2014.

N. Inoue and K. Shinoda, “A Fast and Accurate Video Semantic-Indexing System Using Fast MAP Adaptation and GMM Supervectors,” IEEE Transactions on Multimedia, vol. 14, no. 4-2, pp. 1196-1205, 2012.

N. Inoue, T. Wada, Y. Kamishima, K. Shinoda, and S. Sato, “TokyoTech+Canon at TRECVID 2011,” In Proceedings of the TRECVID Workshop 2011, December 5, 2011.

S. Ji, W. Xu, M. Yang, and K. Yu, “3D Convolutional Neural Networks for Human Action Recognition,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 1, pp. 221-231, January 2013.

Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, “Caffe: Convolutional Architecture for Fast Feature Embedding,” arXiv:1408.5093, 2014.

J. Jin, K. Fu, and C. Zhang, “Traffic Sign Recognition With Hinge Loss Trained Convolutional Neural Networks,” IEEE Transactions on Intelligent Transportation Systems, vol.15, no. 5, pp. 1991-2000, October 2014.

E. R. Kandel, “An Introduction to the Work of David Hubel and Torsten Wiesel,” The Journal of Physiology 587 (Pt 12), pp. 2733–2741, April 2009.

A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, “Large-Scale Video Classification with Convolutional Neural Networks,” In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725-1732, June 2014.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, ‘’Imagenet Classification with Deep Convolutional Neural Networks’’, Advances in Neural Information Processing Systems, pp. 1097-1105, 2012.

I. Laptev and T. Lindeberg, “Space-Time Interest Points,” In Proceedings of International Conference on Computer Vision, pp. 432-439, 2003.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based Learning Applied to Document Recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, November 1998.

L. Lin, C. Chen, M.-L. Shyu, and S.-C. Chen, “Weighted Subspace Filtering and Ranking Algorithms for Video Concept Retrieval,” IEEE Multimedia, vol. 18, no. 3, pp. 32-43, July-September 2011.

L. Lin, G. Ravitz, M.-L. Shyu, and S.-C. Chen, “Video Semantic Concept Discovery using Multimodal-based Association Classification,” In Proceedings of the IEEE International Conference on Multimedia & Expo, pp. 859-862, Beijing, China, July 2-5, 2007.

D. Liu, Y. Yan, M.-L. Shyu, G. Zhao, and M. Chen, “Spatio-temporal Analysis for Human Action Detection and Recognition in Uncontrolled Environments,” International Journal of Multimedia Data Engineering and Management, vol. 6, issue 1, pp. 1-18, 2015.

J. Liu, J. Luo, and M. Shah, “Recognizing Realistic Actions from Videos in the Wild,” In Proceeding of the IEEE International Conference on Computer Vision and Pattern Recognition, pp. 1996-2003, June 20-25, 2009.

J. Liu, Y. Yang, and M. Shah, “Learning Semantic Visual Vocabularies using Diffusion Distance,” In Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition, pp. 461-468, 2009.

X.-Y. Liu, J. Wu, and Z.-H. Zhou, “Exploratory Undersampling for Class-Imbalance Learning,” IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics, vol. 39, issue 2, pp. 539-550, April 2009.

Q. Mao, M. Dong, Z. Huang, and Y. Zhan, “Learning Salient Features for Speech Emotion Recognition Using Convolutional Neural Networks,” IEEE Transactions on Multimedia, vol. 16, no. 8, pp. 2203-2213, December 2014.

T. Meng, Y. Liu, M.-L. Shyu, Y. Yan, and C.-M. Shu, “Enhancing Multimedia Semantic Concept Mining and Retrieval by Incorporating Negative Correlations,” In Proceedings of the 8th IEEE International Conference on Semantic Computing, pp. 28-35, June 16-18, 2014.

V. F. Mota, E. A. Perez, S. M. L. M. Da, M. B. Vieira, and P.-H. Gosselin, “A Tensor Motion Descriptor Based on Histograms of Gradients and Optical Flow,” Pattern Recognition Letters, pp. 85-91, 2014.

P. Over, G. Awad, M. Michel, J. Fiscus, G. Sanders, W. Kraaij, A. F. Smeaton, G. Quenot, and R. Ordelman, “Trecvid 2015 – an overview of the goals, tasks, data, evaluation mechanisms and metrics,” In Proceedings of TRECVID 2015, NIST, USA, 2015.

E. A. Perez, V. F. Mota, L. M. Maciel, D. Sad, and M. B. Vieira, “Combining Gradient Histograms using Orientation Tensors for Human Action Recognition,” In Proceeding of the International Conference on Pattern Recognition, pp. 3460–3463, 2012.

G. Qiong, Z. Li, and C. Zhihua, “Evaluation Measures of the Classification Performance of Imbalanced Data Sets,” In Proceedings of the 4th International Symposium, pp. 461-471, October

2009.

M.-L. Shyu, S.-C. Chen, M. Chen, C. Zhang, and K.Sarinnapakorn, “Image Database Retrieval Utilizing Affinity Relationships,” In Proceedings of the First ACM International Workshop on Multimedia Databases, pp. 78-85, November 7, 2003, New Orleans, Louisiana, USA.

M.-L. Shyu, C. Haruechaiyasak, and S.-C. Chen, “Category Cluster Discovery from Distributed WWW Directories,” Journal of Information Sciences, vol. 155, issues 3-4, pp. 181-197, October 2003.

K. Simonyana and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” arXiv:1409.1556 [cs.CV], 2014.

A. F. Smeaton, P. Over, and W. Kraaij, “Evaluation Campaigns and TRECVid,” In Proceedings of the 8th ACM International Workshop on Multimedia Information Retrieval, pp. 321-330, 2006.

C.G.M. Snoekyz, K.E.A. van de Sandeyz, D. Fontijnez, A. Habibiany, M. Jain, S. Kordumovay, Z. Liy, M. Mazloomy, S.L. Pinteay, R. Taoy, D.C. Koelmayz, and A.W.M. Smeulders, “MediaMill at TRECVID 2013: Searching Concepts, Objects, Instances and Events in Video,” TRECVID 2013, November 26 – 28, 2013.

Y. Sun, T. Osawa, K. Sudo, Y. Taniguchi, H. Li, Y. Guan, and L. Liu, “TRECVid 2013 Semantic Video Concept Detection by NTT-MD-DUT,” TRECVID 2013, November 26–28, 2013.

S. Sural, G. Qian, and S. Pramanik, “Segmentation and Histogram Generation using the HSV Color Space for Image Retrieval,” In Proceeding of the International Conference on Image Processing, pp. 589-592, 2002.

P. Swietojanski, A. Ghoshal, and S. Renals, “Convolutional Neural Networks for Distant Speech Recognition,” IEEE Signal Processing Letters, vol. 21, no. 9, pp. 1120-1124, September 2014.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going Deeper with Convolutions,” arXiv:1409.484, 2014.

C. Unsworth and G. Coghill, “Excessive Noise Injection Training of Neural Networks for Markerless Tracking in Obscured and Segmented Environments,” Neural Computation, vol. 18, no. 9, pp. 2122-2145, September 2006.

D. Verma and V. Maru. “An Efficient Approach for Color Image Retrieval using Haar Wavelet,” In Proceeding of the IEEE International Conference on In Methods and Models in Computer Science, pp. 1–5, 2009.

J. Wan, D. Wang, S.C.H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li, “Deep Learning for Content-Based Image Retrieval: A Comprehensive Study,” In Proceedings of the ACM International Conference on Multimedia, pp. 157-166, Novemember 2014.

Y. Yan, M. Chen, M.-L. Shyu, and S.-C. Chen, “Deep Learning for Imbalanced Multimedia Data Classification,” In Proceedings of the 2015 IEEE International Symposium on Multimedia, pp. 483–488, December 2015.

Y. Yan, J.-W. Hsieh, H.-F. Chiang, S.-C. Cheng, and D.-Y. Chen, “PLSA-Based Sparse Representation for Object Classification,” In Proceedings of the 2014 22nd International Conference on Pattern Recognition, pp. 1295-1300, August 24-28, 2014.

Y. Yan, Y. Liu, M.-L. Shyu, and M. Chen, “Utilizing Concept Correlations for Effective Imbalanced Data Classification,” In Proceedings of the 2014 IEEE 15th International Conference on Information Reuse and Integration, pp. 561-568, August 13-15, 2014.

Y. Yan, S. Pouyanfar, H. Tian, S. Guan, H.-Y. Ha, S.-C. Chen, M.-L. Shyu, and S. Hamid, “Domain Knowledge Assisted Data Processing for Florida Public Hurricane Loss Model,” In Proceedings of the 17th IEEE International Conference on Information Reuse and Integration, pp. 441-447, July 28-30, 2016.

Y. Yan, M.-Ling Shyu, and Q. Zhu, “Supporting Semantic Concept Retrieval with Negative Correlations in a Multimedia Big Data Mining System,” International Journal of Semantic Computing, vol. 10, issue 2, pp. 247-268, 2016.

Y. Yan, M.-Ling Shyu, and Q. Zhu, “Negative Correlation Discovery for Big Multimedia Data Semantic Concept Mining and Retrieval,” In Proceedings of the 10th IEEE International Conference on Semantic Computing, pp. 55-62, February 3-5, 2016.

Y. Yan, Q. Zhu, M.-L. Shyu, and S.-C. Chen, “A Classifier Ensemble Framework for Multimedia Big Data Classification,” the 17th IEEE International Conference on Information Reuse and Integration, pp. 615-622, July 28-30, 2016.

M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica, ”Apache Spark: A Unified Engine for Big Data Processing,” Communications of the ACM, vol. 59, no. 11, pp. 56-65, 2016.

L. Zhang and W. Wang, “A Re-sampling Method for Class Imbalance Learning with Credit Data,” In Proceedings of the 2011 International Conference on Information Technology, Computer Engineering and Management Sciences, pp. 393-397, September 2011.

https://deepmind.com/research/alphago, website of AlphaGo (accessed in 2016)

https://github.com/yahoo/CaffeOnSpark, website of CaffeOnSpark (accessed in 2016)