Fully Convolutional Network (FCN) for Imbalanced and Small Database Classification

Jun Shao

***Abstract*—Classification on small and imbalanced database is difficult for deep neural networks due to limits of samples and the skewed data distribution. In this work, we build and fit a Fully Convolutional Network (FCN) on an industrial small and imbalanced dataset. We test the effects of re-sampling technologies and cost-sensitive loss function in balancing the imbalanced database. Considering the small size of the interested minority classes, we also test a transfer-learning method (i.e., using a pre-trained ResNet to extract features) and a meta-learning architecture on the database. The results show that basic FCN with the ADASYN oversampling in minority classes and Random undersampling in majority class achieves the best performance.**

***Index Terms*—Fully Convolutional Network, Imbalanced Database, Oversampling, Meta-learning.**

I. Introduction

Rare events detection is difficult in industry due to the infrequency of race events. Normally, only insufficient samples can be collected for the interesting events or classes, leading to the problems associated with imbalanced data. Imbalanced data refers to a dataset with one or more classes having a large number of instances while the rest classes with very limited samples, which are usually called minority classes. In the last decade, numerous algorithms have been proposed to solve the imbalance data learning problem. Basically, appraoches can be categorized into two classes: Resampling and cost-sensitive learning [1]. Resampling methods alleviate the skewed data distribution by rebalancing the sample space of the imbalanced dataset. For data resampling, oversampling for the minority class and under-sampling for the majority class are two common practices. The widely used oversampling methods including: Randomly duplicating the minority samples, Synthetic Minority Vver-sampling Technique (SMOTE) [2], Borderline-SMOTE [3] and Adaptive Synthetic Sampling (ADASYN) [4]. While the widely used technology for under-sampling is Random Under-Sampling (RUS) [5].

Cost-sensitive learning [1] assumes that the costs for the misclassification of the minority class should be higher than the loss for majority samples. Although the samples of minority class appear less frequently, the weights for the sample is higher than the majority sample. Compared with re-sampling methods, cost-sensitive method is more computationally efficient without over-sampling minority classes.

Besides the imbalanced learning problem, the application of deep neural network has also been confronted with the overfitting problem caused by small databases. Few-shot learning problem is special case with small database, which usually needs a model to be adapted to new classes not seen in training, while only a few examples (e.g. 10-50 images per class) of these new classes are given. Transfer-learning for feature extraction with a pre-trained deep neural network can be applied to a similar task. But the premise is that a pre-trained network on a similar big database is available for the target task.

Recently, the advent of meta-learning methods shined some light on the few-shot learning problem. Meta-learning, also known as "learning to learn", intends to design models that can learn new skills or adapt to new environments rapidly with a few training examples [6].

In this work, we will test the methods mentioned above for a wood knot classification task on a small and imbalanced database.

Jun Shao is with the Department of Computer Science and Software Engineering, Concordia University, Montreal, Quebec, H3G 2W1, Canada (e-mail: shao12jun@gmail.com).

TABLE I
Fully Convolutional Network Architecture ('K','S' and 'P' denote 'Kernel size','stride' and 'padding').

	Layer	Output Size	Details
Input	Conv1	$115 \times 115 \times 32$	K3, S1, P1
Hidden Layer	Conv2	$57 \times 57 \times 64$	K3, S1, P1
	Conv3	$28 \times 28 \times 128$	K3, S1, P1
	Conv4	$14 \times 14 \times 256$	K3, S1, P1
	Conv5	$7 \times 7 \times 512$	K3, S1, P1
	Conv6	$3 \times 3 \times 1024$	K3, S1, P1
	Conv7	$1 \times 1 \times 100$	K1, S1, P0
	Conv8	$1 \times 1 \times 3$	K3, S1, P0
Output	Softmax	$1 \times 1 \times 3$	—-

Note: Convolutions (1-6) are followed by BatchNorm, ReLU and MaxPool(2).

II. Proposed Method

A. Fully Convolutional Network

Fully Convolutional Network (FCN) [7] is an architecture which is built only from locally connected layers, such as convolution, pooling and up-sampling. No fully connected layer is used in FCNs, thus reduces the number of parameters and computation time. Besides, FCN can work regardless of the original image size, without requiring any fixed number of units at any stage, given that all connections are local.

B. Re-sampling and Cost Sensitive Loss

In this work, we compared the effects of re-sampling with cost-sensitive loss for alleviating the impacts of imbalanced data distribution. For Re-sampling, we adopt oversampling the minority classes plus under-sampling the majority class [2]. The oversampling technologies include (SMOTE) [2], and ADASYN [4]. SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors.

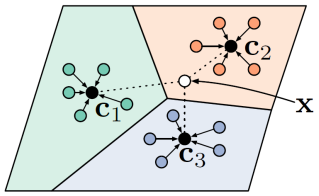

Fig. 1. Prototypical networks for few-shot learning. Few-shot prototypes $\mathbf{c}_k$ are computed as the mean of embedded support examples for each class. Then embedding query points are classified via softmax over distances to class prototypes.The figure is from [11].

The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b [8]. ADASYN is based on the idea of adaptively generating minority data samples according to their distributions: more synthetic data is generated for minority class samples that are harder to learn compared to those minority samples that are easier to learn [4].For the under-sampling, we utilize the Random Under-sampling.

For cost-sensitive loss we choose the CrossEntroyLoss() as the loss function and set the $weight = [0.94, 0.12, 0.94]$ according to the size of each class in the database during training.

C. Transfer-Learning With Pre-trained Deep Neural Network

To make use of the existing big databases and models, we adopt the pre-trained ResNet [9] with 18 convolutional layers, to extract features for our Network. ResNet18 was well trained in ImageNet [10]. We fix the parameters of the convolutional layers and drop out the last non-convolutional layers. After that, the extracted features with size of (512,4,4) was fed into a four layers of FCNs.The first two convolutions with the parameters: K=3, S=1 and P=1, are followed by BatchNorm, MaxPool and ReLU. The last two convolutions are same as the corresponding layers in the basic FCNs above with parameter: K=3, S=1 and P=0.

In this work, we only adopt ADASYN+Random Under-sampling for the Transfer-learning model.

D. Meta-Learning

Meta-learning is also called "learning to learn". The method requires a classifier must be able to generalize to new classes not seen in the training set, given only a small number of examples of the new classes. In this work, We adopt a feature distance based method, called Prototypical networks [11] for meta-learning. Prototypical networks learn a metric space (e.g. Euclidean Distance of Cosine similarity) in which classification can be performed by computing distances to prototype representations of each class.

As shown in Fig. 1,the model at first learns an encoder f_θ, which transfers the supported samples to the prototype (embedding) of the class. Given the distance function d, prototypical networks produce a distribution over classes for a query point x based on a softmax over distances to the prototypes in the embedding space:

$$\mathcal{P}_\theta(y = k|x) = \frac{exp(-d(f_\theta(X), \mathbf{c}_k))}{\sum_{k'} exp(-d(f_\theta(x)), \mathbf{c}_{k'})} \tag{1}$$

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(x_i, y_i) \in S_k} f_\theta(x_i) \tag{2}$$

Where S_k is the support set of class k.

TABLE II
THE STRUCTURE OF THE PLYWOOD KNOTS DATABASE.

Class	LargeKnots	NoDefect	SmallKnots	Total
Train	56	917	63	1036
Val	8	131	8	147
Test	16	262	16	294
All	80	1310	87	1477

E. Evaluation Metrics

Considering the imbalance of the database, we use F1 measure as the evaluation metric for our model. The formula for F1 is shown as following:

$$\mathbf{F}1 = \frac{2P * R}{P + R} \tag{3}$$

$$P = \frac{TP}{TP + FP}, R = \frac{TP}{TP + FN} \tag{4}$$

Where P, R is precision of the target class, while TP, FP and FN is Ture Positive, False Positive and False Negative samples, respectively.

III. EXPERIMENTS

A. Dataset

We train and evaluate our FCNs on the plywood knots database provided by Matrox. The database consists of three classes i.e. LargeKnots, NoDefect and SmallKnots. The size of the database is 1477 with two minority classes (LargeKnots: 80, SmallKnots: 87) and one majority class (NoDefect:1310). We randomly split the database into training, validation and test set as 7:1:2. The structure of the database before and after splitting is shown in Table II.

For the meta-learning experiment, we combine the training and validation set above as one new validation set for the meta-learning training. We randomly choose 100 classes from the ImageNet ILSVRC dataset[1].

B. Implementation Details

For the basic FCNs and ResNet+FCNs experiments, we utilize Adam [12] for model optimization with the following hyper-parameters: learning rate = 0.0001, beta1=0.9, beta2=0.999, and batch size=64. We train the models for 100 epochs with weight decay of 0.5 for every 20 epochs. We save the best model according to the highest average F1 value of the

[1]The data is downloaded from http://www.robots.ox.ac.uk/∼vgg/decathlon/.

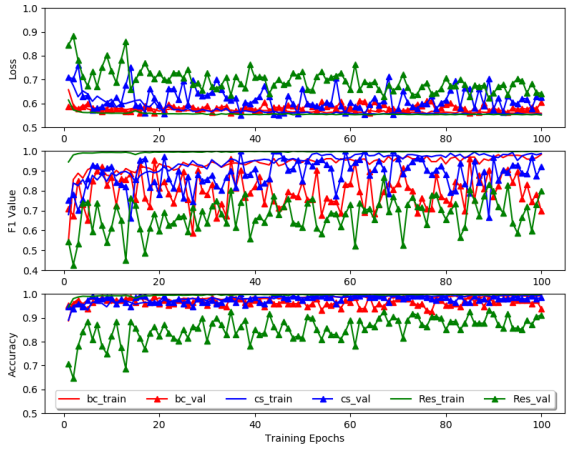

Fig. 2. Training process of basic FCN, FCN+cost-sensitive loss and FCN+SMOTE model.

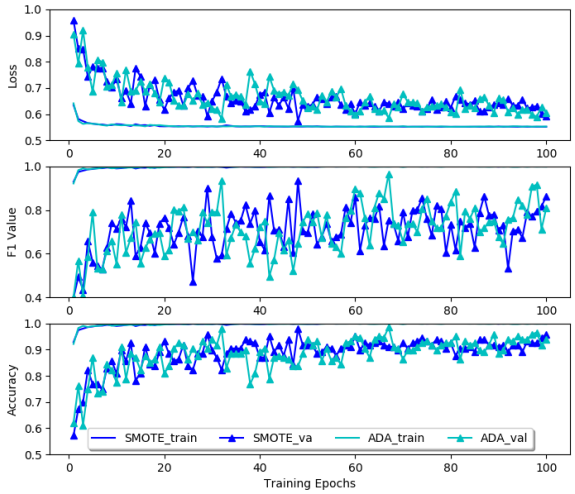

Fig. 3. Training process of FCN+SMOTE and FCN+ADASYN model.

validation experiments. The FCNs are trained on a NVIDIA GeForce GTX 1080 Ti for 5 15 min.

The code for our prototypical meta-learning experiment is adapted from [11]. The parameters for system training are: train.way=60, train.shot=5, train.query=5, test.way=3,test.shot=5, test.query=15, train.episode=100. We train the system for 2000 epochs with early stopping (the patience is 1000). The test parameters are: train.way=3, train.shot=5, train.query=15, test.way= 3,test.shot=5, test.query=11, test.episode=100. The model is trained on the same GPU for about 3h.

TABLE III
THE COMPARISON OF MODEL PERFORMANCE AMONG FCNS AND RESNET+FCNS (C1,C2,C3 ARE THREE CLASSES).

Model	Test F1				Test Accuracy	Test time(s)
	Average	C1	C2	C3		
Basic FCN	0.921	0.867	0.994	0.903	0.983	3.00
Basic FCN+ cost sensitive	0.958	0.968	0.996	0.909	0.990	3.14
Basic FCN+ SMOTE	0.968	0.968	0.998	0.938	0.993	3.05
Basic FCN+ ADASYN	0.978	1.000	0.997	0.938	0.993	3.26
Res+FCN+ ADASYN	0.933	0.875	0.992	0.933	0.983	11.82

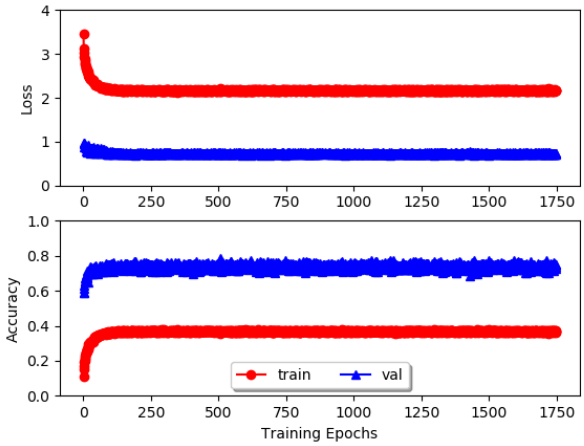

Fig. 4. The change of the training and validation loss and accuracy during the raining process of prototypical network.

C. Experimental Results of FCNs and Transfer-learning

The training/validation loss, average F1 value and accuracy of five experiments are shown in Fig. 2 and Fig. 3. We can find that due to the small size of validation set, the validation loss, f1 value and accuracy vibrate obviously. While training indicators change smoothly because of the relatively bigger size of training set. Although the database we trained is small, we did not observe obvious over-fitting during the training progress, because there is not an obvious drop of validation f1 value during the later stage of training.

As shown in Table III, FCN models with oversampling methods achieve the two highest average f1 value in test, 0.979 and 0.968 for ADASYN and SMOTE,respectively, which shows the superiority of oversampling method for alleviating the impacts of imbalanced distribution. FCN combined with cost-sensitive loss outperforms the basic FCN without any methods for data imbalance, but is prior to the oversampling methods. Compared with FCN+ANASYN, deep neural network(ResNet+FCN model) does not show their superiority on small database.

Compared with Deep neural network (ResNet+FCN), the FCN based models achieve a faster test speed 3.00-3.26 seconds (0.01 frame/sec) than ResNet based model(0.04 frame/sec).

D. Experimental Results of Meta-Learning

As shown in Fig. 4,there is no obviously over-fitting during the training process. However, the meta-learning on the Im-

ageNet ILSVRC dataset seems to make little contribution to the learning on plywood knots database. We test the learned meta-learning system on plywood knots database, and the test accuracy is 0.786. The relatively lower performance of Prototypical network may be caused by the random classes we selsected from the ImageNet ILSVRC dataset. In the ImageNet it is hard to select some classes that are similar with wood knots. The algorithms chosen for meta-learning also impact the performance.

Although meta-learning achieves a lower performance than other FCN based networks in this work, meta-learning is a promising methods for few-shot learning and even zero-shot learning. When the classes increase, the performance of FCN may decrease rapidly, while meat-learning is robust to the number of classes to test.

IV. Conclusion

In this work, we build a Fully Convolutional Network (FCN) for wood knots classification. Due to the small size of the database and imbalanced data distribution, we tested and compared the methods of re-sampling, cost-sensitive loss, feature extraction by a pre-trained ResNet model and meta-learning system on the small and imbalanced database. Extensive experiments show that over-sampling(ADASYN)+RandOm Undersampling achieve the best performance than other methods. However, meta-learning is still a promising method for classification with more classes and fewer samples.

References

[1] G. Haixiang, L. Yijing, J. Shang, G. Mingyun, H. Yuanyue, and G. Bing, "Learning from class-imbalanced data: Review of methods and applications," *Expert Systems with Applications*, vol. 73, pp. 220–239, 2017.

[2] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[3] H. Han, W.-Y. Wang, and B.-H. Mao, "Borderline-smote: a new over-sampling method in imbalanced data sets learning," in *International conference on intelligent computing*. Springer, 2005, pp. 878–887.

[4] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *2008 IEEE international joint conference on neural networks (IEEE world congress on computational intelligence)*. IEEE, 2008, pp. 1322–1328.

[5] M. A. Tahir, J. Kittler, K. Mikolajczyk, and F. Yan, "A multiple expert approach to the class imbalance problem using inverse random under sampling," in *International workshop on multiple classifier systems*. Springer, 2009, pp. 82–91.

[6] L. Weng, "Meta-learning: Learning to learn fast." [Online]. Available: https://lilianweng.github.io/lil-log/2018/11/30/meta-learning.htmldefine-the-meta-learning-problem

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[8] H. He and Y. Ma, *Imbalanced learning: foundations, algorithms, and applications*. John Wiley & Sons, 2013.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] L. Fei-Fei, "Imagenet: crowdsourcing, benchmarking & other cool things," in *CMU VASC Seminar*, vol. 16, 2010, pp. 18–25.

[11] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," in *Advances in neural information processing systems*, 2017, pp. 4077–4087.

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.