Supervised Two-Stage Transfer Learning on Imbalanced Dataset for Sport Classification

Tianyu Bi[1,2](✉), Dmitri Jarnikov[1], and Johan Lukkien[1]

[1] Department of Mathematics and Computer Science, Eindhoven University of Technology, P.O. Box 513, 5600 Eindhoven, MB, The Netherlands
{t.bi.1,d.s.jarnikov,j.j.lukkien}@tue.nl

[2] Irdeto, High Tech Campus 84, 5656 Eindhoven, AG, The Netherlands

Abstract. Sport classification is a crucial step for content analysis in a sport stream monitoring system. Training a reliable sport classifier can be a challenging task when the data is limited in amount and highly imbalanced. In this paper, we introduce a supervised two-stage transfer learning (Two-Stage-TL) method to solve the data shortage problem. It can progressively transfer features from a source domain to the target domain using a properly selected bridge domain. For the class imbalance issue, we compare several existing methods and demonstrate that the log-smoothing class weight is the most applicable way for this specific problem. Extensive experiments are conducted using ResNet50, VGG16, and Inception-ResNet-v2. The results show that Two-Stage-TL outperforms classical One-Stage-TL and achieves the best performance using log-smoothing class weight. The in-depth analysis is useful for researchers and developers in solving similar problems.

Keywords: Multimedia content analysis · Sport classification · Transfer learning · Class imbalance learning

1 Introduction

The online piracy of media content is widespread, especially for sport streams. To combat piracy of sport streams, content protection companies usually apply a sport stream monitoring system. Figure 1 shows the workflow of such a system, which tracks illegal activities, detects pirated sport streams, collects sample data for sport classification and content identification, and finally enforces content rights. With respect to sport classification, traditional systems either need human in the loop, which is not cost-effective and not scalable, or use handcrafted classifiers, which are not robust and lack flexibility. Thus, there is a need to build a better sport classification model for the system.

Automatic sport classification is a sub-topic of multimedia content analysis. Existing approaches can be categorized by the type of data (single image or images/video), or by the algorithms (handcrafted features or deep learning) as shown in Table 1.

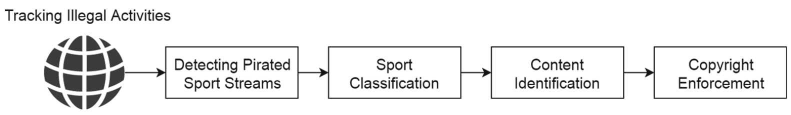


Fig. 1. Workflow of a sport stream monitoring system. For each sport stream, several screenshots are collected as the sample data for further analysis.

Table 1. Existing methods.

	Handcrafted features	Deep learning
Single image	Fisher Kernel [1], Bag-of-features [2]	Deep CNN image classifiers [3–5]
Images/video	Sparse features [6], Dense trajectory [8]	Two-stream [7], C3D [9], LSTM [10]

Since the data collected by the system is single image, we focus on image classification methods. Deep Convolutional Neural Networks (CNN) such as ResNet50 [4] and Inception-ResNet-v2 [5] have shown state-of-the-art performance and outperformed the handcrafted methods. Thus, we decide to build a sport image classification model based on deep learning methods.

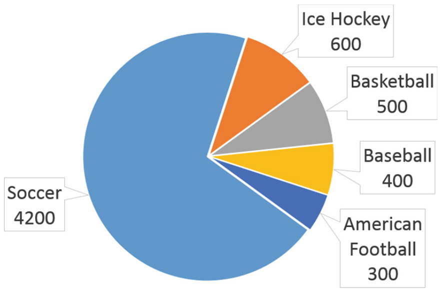


Fig. 2. Pie chart of Live-Sports dataset.

To train our model, we collect a dataset using the sport stream monitoring system and manually label the data. This dataset is called Live-Sports, which has five sport categories as shown in Fig. 2. Due to the dataset limitation, we are facing two challenges. The first challenge is the data shortage problem. Training deep learning models usually require massive amounts of data. However, we only have 6000 images over five sport categories. By training with a limited amount of data, the generalization error of the supervised learning model can be high [11]. The second challenge is the class imbalance problem. As shown in Fig. 2, the dataset is highly imbalanced in categories. Recent study [12] reveals that the class imbalance problem can have a detrimental effect on classification performance especially when the class imbalance ratio is large. Since our target data has a large imbalance ratio, class balancing methods need to be applied to reduce the detrimental effect on the target performance.

To address the first challenge, supervised transfer learning with fine-tuning, can be applied, which reuses the deep learning model or features trained on one task to another related task [13]. For general image classification tasks, a representative way is to use deep learning models that are pretrained on ImageNet and fine-tuned on the target domain [11]. It can be called one stage transfer learning (One-Stage-TL), which transfers features from one source to one target. One-Stage-TL works well for most cases since the earlier layers in the pretrained network extract general features such as edges, colors, and textures, which have high transferability for general image classification tasks [14]. However, the latest findings also reveal that feature transferability drops considerably in the higher layers when there is a large discrepancy between the source and target domains [11]. Thus, One-Stage-TL may not be enough for achieving optimal transferability.

In this work, we suggest an intuitive method that is called supervised two-stage transfer learning (Two-Stage-TL). It establishes knowledge transfer from a source domain to a target domain by using a bridge domain. In this way, it keeps the general features in lower layers, and at the same time, transfers task-related features in higher layers. Feature transferability is enhanced by gradually reducing the domain discrepancy in two stages. Similar ideas have been shown in [15–17]. Our Two-Stage-TL approach is different from theirs as it is designed for deep learning models and it uses a two-step fine-tuning scheme, which fully fine-tunes the CNN model in each step. Based on the defined properties in Sect. 2.1, we find that ImageNet is a good choice for source domain and Sports-1M [18] is a suitable bridge domain. In the experiment, we compare Two-Stage-TL with One-Stage-TL and other training methods using ResNet50 as the model architecture for the sport classification task. We also evaluate the performance of Two-Stage-TL using VGG16 and Inception-ResNet-v2 for comparison. Experimental results show that Two-Stage-TL always achieves better performance than the common One-Stage-TL.

To address the second challenge, existing solutions include oversampling [12], undersampling [12], class weight [12], and imbalance fine-tuning [19]. In this paper, we use a log-smoothing class weight method and compare it with existing methods mentioned above. Experimental results show that the sport classification model achieves the best performance when applying log-smoothing class weight with Two-Stage-TL.

Our contribution consists of three parts. First, we demonstrate that for multi-class classification with a limited number of training data, the Two-Stage-TL method outperforms the One-Stage-TL method if a proper bridge domain is selected. Second, we compared several existing methods for the class imbalance problem and demonstrated that for this specific problem, the log-smoothing class weight is the best way to reduce the impact of class imbalance. Furthermore, extensive experiments are conducted on different CNN models to find the optimal solution considering the tradeoff of accuracy, training time, and model size. Finally, since data shortage and class imbalance are common problems, our in-depth problem analysis and solution are not limited to a specific application and could be helpful to solve similar problems in other applications.

2 Two-Stage Transfer Learning

2.1 Two-Stage Transfer Learning Using Bridge Domain

For image classification tasks, researchers always recommend to pretrain the model on a large-scale publicly available dataset and then fine-tune it on the target dataset. This approach, which is called fine-tuning, has been widely-used for supervised learning tasks. In this paper, we call it one-stage transfer learning (One-Stage-TL), which only transfers knowledge once from a source task to the target task. Here, the source domain and target domain can be denoted by $D_s = (x_i^s, y_i^s)_{i=1}^{n_s}$ and $D_t = (x_j^t, y_j^t)_{j=1}^{n_t}$ respectively, where x_i^s and x_j^t are training samples, y_i^s and y_j^t are labels, and n_s and n_t are the number of samples.

One-Stage-TL can improve the performance on the target task when the source is similar to the target. However, when the source data is quite different, it may lead to very limited performance improvement on the target task due to the low feature transferability. To further improve the target performance, we introduce a very intuitive supervised two-stage transfer learning (Two-Stage-TL) approach. Different from One-Stage-TL, it progressively transfers knowledge from the source to the target by using a bridge domain in the middle. The bridge domain can be denoted by $D_b = (x_k^b, y_k^b)_{k=1}^{n_b}$, where x_k^b represents the training sample, y_k^b represents the label, and n_b is the number of samples.

To guarantee the effectiveness of Two-Stage-TL, the bridge domain should have certain properties. Based on practical experience, we make some assumption with respect to the properties of the bridge domain. First, compared with the task of source t_s, the task of the bridge t_b should be more related to the task of the target t_t. Second, the data distribution of the bridge D_b should be more similar to the target distribution D_t than the source distribution D_s. Third, the bridge dataset X_b should be larger than the target dataset X_t. Finally, since Two-Stage-TL is used for supervised learning tasks, the bridge domain should have labeled data without heavy cleaning work.

Based on our assumption, we find Sports-1M can be a good bridge domain given ImageNet as the source and Live-Sports as the target. Sports-1M [18] is a publicly available dataset, which has approximately 1 million YouTube video links for 487 sport categories. We collect a dataset of Sports-1M with the five sports of interest. We collect 2000 frames extracted from about 100 videos for each sport. Non-sports contents such as commercials or interviews are removed in advance. We find that the dataset is a hybrid of professional sports, user generated contents and remix, while the target dataset contains only professional sports data. Thus, Sports-1M has the same task, and visually different but very similar data compared with the target domain, which meets the requirements of the bridge domain.

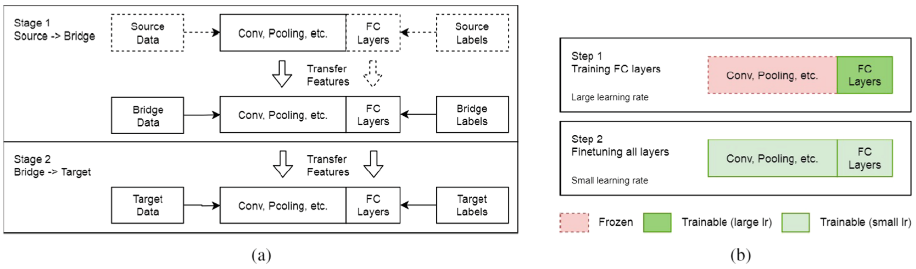


Fig. 3. Two-Stage-TL framework: (a) The Two-Stage-TL approach. Source data is not required if the pre-trained model is available. (b) The two-step fine-tuning scheme. When the target dataset is small, a smaller learning rate is applied in step 2 to avoid overfitting [13].

2.2 Two-Stage-TL Framework

The framework of our Two-Stage-TL approach is shown in Fig. 3. In stage 1, we pretrain the model on the source domain, transfer the features, and fine-tune the model on the bridge domain. To speed up the pretraining step, we can use off-the-shelf features that are pretrained on ImageNet or other benchmarks. In this case, we do not need to collect data and train the model for the source domain. If the bridge has a different task, the model needs modifications on fully-connected (FC) layers. In this case, when we transfer the features from source to bridge, features on FC layers can remain for fine-tuning or be replaced by random initialization. Since FC layers are task-specific with a larger transferability gap [11], it should be fully trained on the new task, while the lower layers, which contain general features, should be gently fine-tuned to further improve the performance [13]. Thus, we use a two-step fine-tuning scheme as shown in Fig. 3(b). In this scheme, the model is trained on FC layers with a large learning rate in step 1 and fine-tuned on all layers with a smaller learning rate in step 2. After training the bridge model, we transfer features of all layers to the target model and fine-tune the model on the target task in stage 2. Similarly, the target model is fine-tuned using the two-step fine-tuning scheme.

2.3 Class Imbalance Learning

We evaluate four class balancing methods including oversampling, undersampling, class weight, and imbalance fitting. Oversampling is a widely-used sampling method proven to be effective in many situations [12]. In our experiment, we choose random minority oversampling, which randomly selects samples from minority classes and applies data augmentation. Undersampling is another sampling method, which is preferable to oversampling in some cases [12]. We choose random majority undersampling that removes randomly selected samples from majority classes.

Class weight is another common approach, which assigns different loss for different classes by giving higher weight to the minority class and lower weight to the majority class. The original class weight is calculated by the equation: $cw_i = n_{majority}/n_i$, where cw_i denotes the class weight for class i, $n_{majority}$ is the number of samples for the majority class, and n_i is the number of samples for class i. Since our dataset is highly imbalanced, the class weight of a minority class can be very large. In this case, we need to smooth the class weight to avoid getting biased on the minority class due to the large weight. Simply dividing the original class weight by a constant value is not enough, since the majority class suffers from a too small weight. We introduce a log-smoothing method that smooths the class weight by a natural logarithm function as follows: $cw_i^{ln} = ln(cw_i) + 1$. In this way, the class weight of minority classes shrinks to a reasonable level, while the class weight of the majority class keeps the same. Both default and log-smoothing class weight methods are evaluated in the experiment.

The last method, which we call imbalance fitting, is inspired by the method in [19]. In our implementation, we first train the network on the balanced data (by undersampling), then fully fine-tune the network with the original dataset. To find the optimal method for our target dataset, we evaluate all of the above methods using different training approaches.

3 Experiments

3.1 Dataset

In our experiment, ImageNet is the source domain and the pretrained features are used for transfer learning in stage 1. Live-Sports and Sports-1M are the target domain and bridge domain. They both have five sport categories: American Football, Baseball, Basketball, Ice Hockey, and Soccer. Live Sports has 6000 images: 100 images for validation, 100 images for testing for each category, and the rest are used for training. Sports-1M has 12500 images: 2000 images for training and 500 images for validation for each category.

For the undersampling method, we create a balanced training set of Live-Sports and each class has the same number of images (100) as the minority class (American Football). The balanced samples are randomly selected from the original training set. For the oversampling method, a balanced training set is created by using data augmentation. Each category has the same number of images (4000) as the majority class (Soccer).

3.2 Experimental Environment and Settings

The experimental environment is a PC with an Intel Xeon E5 CPU and an Nvidia Tesla V100 GPU with 32 GB of memory. We select ResNet50, VGG16, and Inception-ResNet-v2 as the basic CNN models in our experiment, because they are widely used for image classification and have different levels of depth and size. All models used in the experiments are implemented using Keras with TensorFlow as the backend.

To preprocess training and testing data, we resize the images to a certain size, which is $224 * 224$ for ResNet50 and VGG16 and $299 * 299$ for Inception-ResNet-v2. For real-time data augmentation during training, we use several image processing methods provided by Keras ImageDataGenerator. The image processing methods include rotation ($-90°..90°$), width shift ($-20\%..20\%$), height shift ($-20\%..20\%$), shear ($-0.2°..0.2°$), zoom ($-20\%..20\%$), horizontal flip (*prob* 50%), and vertical flip (*prob* 50%).

We use stochastic gradient descent (SGD) with momentum as the optimization strategy in our experiment. The momentum is set to 0.9, the batch size is set to 32, and the initial learning rate is set to 0.01. For the second step of the two-step fine-tuning scheme, which fine-tunes on all layers, we use a smaller learning rate (0.001) to avoid overfitting to the target domain [13]. For training from scratch, the model is trained by 100 epochs. For transfer learning, the model is trained by 50 epochs in step 1 and 50 epochs in step 2. Instead of running through all the epochs, we stop training when the validation loss does not improve in 10 epochs.

In the test phase, we use classification accuracy and training time as the evaluation metrics.

3.3 Comparison of Different Training Approaches

In this section, we evaluate the performance of Two-Stage-TL and other training methods including Train-From-Scratch, Train-From-Scratch-NoAug (without data augmentation), and One-Stage-TL (using ImageNet as source). Classification accuracy and training time are used to evaluate these methods. Table 2 shows the evaluation results using ResNet50 as the basic network.

Table 2. Classification accuracy and training time of different training approaches.

Approaches	Accuracy	Time (min)
Train-From-Scratch-NoAug	77.8%	19
Train-From-Scratch	79%	35
One-Stage-TL	90.4%	54
Two-Stage-TL	93%	124 + 63

From Table 2, we can see that Train-From-Scratch-NoAug has the lowest classification accuracy, which is only 77.8%. The classification accuracy of Train-From-Scratch increases by 1.2% because of data augmentation. However, it is still quite low (79%), which shows that training from scratch only is not enough for training a reliable sport classifier. One-Stage-TL achieves much higher performance, which is 90.4%. It shows that the pretrained weights on ImageNet are beneficial for our task, sport classification. Compared with other methods, Two-Stage-TL achieves the highest classification accuracy, which is improved by 2.6%

from One-Stage-TL. It demonstrates that using Sports-1M as a bridge between ImageNet and the target can further improve the classification accuracy. The training time of Two-Stage-TL is more than other methods, which is 124 min in the first stage and 63 min in the second stage. The time of the first stage can be ignored, since it is only conducted once and the weights can be reused in the future.

3.4 Comparison of Different Class Balancing Methods

In this section, we evaluate the performance of different class balancing methods including oversampling, undersampling, class weight, and imbalance fitting (Imb-Fit). We use ResNet50 for this experiment because it has a good trade-off between high performance and low training time. From Table 3, we find that oversampling has an enhancement in performance for all training approaches. Compared with other class balancing methods, oversampling enables the highest classification accuracy for Train-From-Scratch and One-Stage-TL. The performance for Train-From-Scratch and One-Stage-TL approaches with the undersampling method drops by 5.2% and 1% respectively compared with the original performance (in Table 2). The reason can be that removing training examples in undersampling affects the generalizability on the test set. For Two-Stage-TL, the undersampling method achieves higher performance than the original setting and oversampling. The default class weight method does not work well on Train-From-Scratch because the model cannot converge under the higher training loss. Two-Stage-TL with log-smoothing class weight achieves better performance, which is 1% higher than using the default class weight method. The imbalance fitting method does not improve the best performance of any training approach. Overall, Two-Stage-TL with log-smoothing class weight is considered as the most applicable approach for our problem.

Table 3. Classification accuracy of different class balancing methods (* log-smoothing).

Approaches	Undersampling	Oversampling	Class weight	Imb-Fit
Train-From-Scratch	72.6%	82.8%	20%	78.6%
One-Stage-TL	89.4%	93.2%	91.6%	91.6%
Two-Stage-TL	94%	93.4%	93% (*94%)	94%

Additionally, we compare the training time of the training approaches with different class balancing methods. From Fig. 4, we can see that for most cases oversampling has the longest training time while undersampling has the shortest training time. This is caused by the different size of the training set, which is 500 for undersampling and 20000 for oversampling. Imbalance fitting and class weight require medium level training time. If training time is crucial, Two-Stage-TL with undersampling is the most applicable one, even though it may lose useful information from training data.

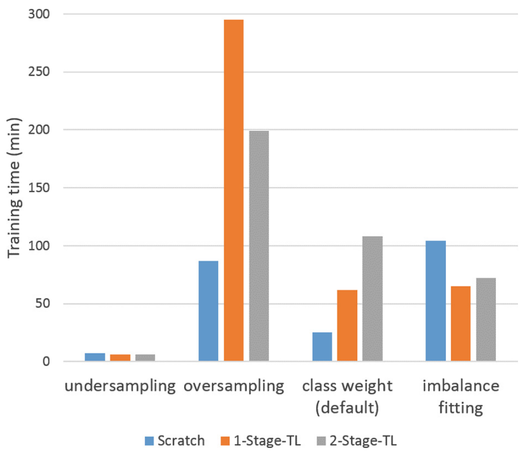


Fig. 4. Training time.

3.5 Comparison of Different CNN Models

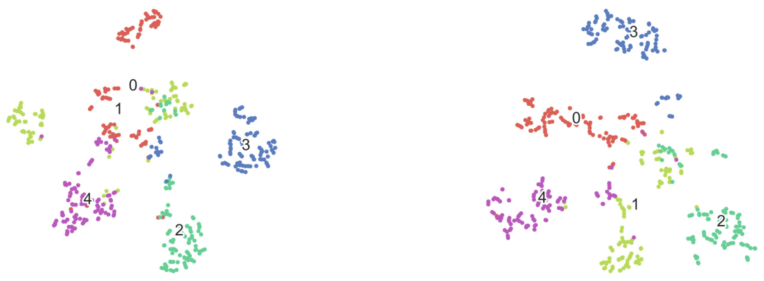
In this section, we compare the performance of Two-Stage-TL on different deep CNN models including ResNet50, VGG16, and Inception-ResNet-v2. We use undersampling as the class balancing method for all of them. We evaluate the models using classification accuracy and also compare the model size and the training time in two stages. Table 4 shows the classification accuracy, training time, and model size of three deep CNN models. We find that Inception-ResNet-v2 achieves the best accuracy (96.8%), while ResNet50 achieves a bit lower accuracy (94%) but requires much less training time (136 min) and has much smaller model size (196 MB). For practical implementation, if there is a limitation for training time and model size, ResNet50 is a good choice. Inception-ResNet-V2 is optimal when the classification accuracy is crucial.

Table 4. Classification accuracy, training time, and model size of different CNN models. T1 and T2 refer to the training time in two stages.

Approaches	Accuracy	T1 (min)	T2 (min)	Size (MB)
VGG16	93.8%	270	13	968
ResNet50	94%	130	6	196
Inception-ResNet-v2	96.8%	329	11	428

3.6 Feature Visualization

To demonstrate Two-Stage-TL has better transferability than One-Stage-TL, we visualize Two-Stage-TL features and One-Stage-TL features of test images. The features are extracted from the last hidden layer of the ResNet50 models trained by Two-Stage-TL and One-Stage-TL. We use the t-SNE method to



(a) One-Stage-TL (b) Two-Stage-TL

Fig. 5. Visualization of t-SNE embeddings

reduce dimensions of the features and plot the t-SNE embeddings in a 2D space for visual analysis. The t-SNE embeddings of Two-Stage-TL and One-Stage-TL features are shown in Fig. 5, in which the data points of the same class are drawn in the same color. Our observation is that the test examples with Two-Stage-TL features are discriminated better compared with One-Stage-TL features. The samples of each class in Two-Stage-TL features are better clustered with clearer boundaries. The observation implies that Two-Stage-TL improves the transferability of the features to the target domain. The finding can explain the better performance of Two-Stage-TL over One-Stage-TL.

4 Conclusion

In this paper, we introduced a supervised two-stage transfer learning (Two-Stage-TL) method, which improves feature transferability by reducing the domain discrepancy progressively. To verify its effectiveness, we conducted experiments using three deep CNN models: ResNet50, VGG16, and Inception-ResNet-v2. To solve the class imbalance problem, we evaluated different class balancing methods. The experimental results show that the Two-Stage-TL outperforms the classical One-Stage-TL, and it achieves the best performance using together with log-smoothing class weight. In future work, we will extend Two-Stage-TL to Multi-Stage-TL and explore its feasibility in multi-model applications.

References

1. Perronnin, F., Sánchez, J., Mensink, T.: Improving the Fisher Kernel for large-scale image classification. In: Daniilidis, K., Maragos, P., Paragios, N. (eds.) ECCV 2010. LNCS, vol. 6314, pp. 143–156. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15561-1_11
2. Nowak, E., Jurie, F.: Sampling strategies for bag-of-features image classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 490–503. Springer, Heidelberg (2006). https://doi.org/10.1007/11744085_38

3. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
4. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
5. Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A.: Inception-v4, inception-resnet and the impact of residual connections on learning. In: Thirty-First AAAI Conference on Artificial Intelligence, February 2017
6. Dollár, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: VS-PETS, Beijing, China, October 2005
7. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems, pp. 568–576 (2014)
8. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 3551–3558 (2013)
9. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3D convolutional networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 4489–4497 (2015)
10. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4694–4702 (2015)
11. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. arXiv preprint arXiv:1502.02791 (2015)
12. Buda, M., Maki, A., Mazurowski, M.A.: A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw. **106**, 249–259 (2018)
13. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks?. In: Advances in Neural Information Processing Systems, pp. 3320–3328 (2014)
14. Ghazi, M.M., Yanikoglu, B., Aptoula, E.: Plant identification using deep neural networks via optimization of transfer learning parameters. Neurocomputing **235**, 228–235 (2017)
15. Pan, S.J., Kwok, J.T., Yang, Q.: Transfer Learning via dimensionality reduction. In: AAAI, vol. 8, pp. 677–682, July 2008
16. Shan, W., Sun, G., Zhou, X., Liu, Z.: Two-stage transfer learning of end-to-end convolutional neural networks for webpage saliency prediction. In: Sun, Y., Lu, H., Zhang, L., Yang, J., Huang, H. (eds.) IScIDE 2017. LNCS, vol. 10559, pp. 316–324. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-67777-4_27
17. Kim, H.G., Choi, Y., Ro, Y.M.: Modality-bridge transfer learning for medical image classification. In: 2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), pp. 1–5. IEEE, October 2017
18. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1725–1732 (2014)
19. Havaei, M., et al.: Brain tumor segmentation with deep neural networks. Med. Image Anal. **35**, 18–31 (2017)