Improving Chairlift Security with Deep Learning

Kevin Bascol[1,2] & Rémi Emonet[1] & Elisa Fromont [1] & Raluca Debusschere[2]

[1] Univ Lyon, UJM, CNRS, Lab Hubert Curien UMR5516, F-42000
[2] Bluecime inc., 445 rue Lavoisier, 38330 Montbonnot Saint Martin, France

Abstract. This paper shows how state-of-the-art deep learning methods can be combined to successfully tackle a new classification task related to chairlift security using visual information. In particular, we show that with an effective architecture and some domain adaptation components, we can learn an end-to-end model that could be deployed in ski resorts to improve the security of chairlift passengers. Our experiments show that our method gives better results than already deployed hand-tuned systems when using all the available data and very promising results on new unseen chairlifts.

Keywords: Deep learning, Convolutional neural networks, Image Classification, Domain adaptation.

1 Introduction

Ski resorts are all equipped with different and numerous chair and ski-lifts. To ensure the security at each terminal, one person (at least) is in charge of continuously monitoring the lift traffic and ensuring safe boarding for all its passengers. Possible endangering situations include: disorderly waiting queue, improper seating, restraining bar not pulled down, loss of a gliding equipment (ski, snowboard), unaccompanied child, etc. To ease this tedious work, all the chairs could be equipped with sensors that could monitor the number of persons on a given chair, whether they are well seated, whether the position of the restraining bar (or security railing) is correct a few seconds after leaving the boarding station (is very important to prevent falls), whether a child is not alone on the chair, whether the skiers have not lost a ski or a snowboard etc. Such sensors should trigger an alarm that would stop or slow down the chairlift if a particular anomaly is detected. However, in the extreme weather conditions that ski-resorts can face, small mechanical sensors are subject to wear and often prone to failure, thus other more robust solutions are expected.

The work presented in this article is made in collaboration with a start-up called Bluecime which has successfully shown that video cameras represent a more reliable solution and that the real-time image processing of the captured images leads to better results for the cited tasks than mechanical sensors. The company has installed video cameras filming the boarding platforms in several ski-resorts. Each camera is linked to a computer that processes the video stream in real time. They have developed a software that takes as input the video of a

chair moving on the chairlift, focuses on the chair and automatically (using signal processing techniques) detects if the security railing is up or down a few seconds after boarding. The results obtained by the company are already very good but they are currently limited to one particular task (triggering an alarm if the restraining bar position is not as expected). Moreover, in some processing units, their system needs to be calibrated on a per-lift basis, which is time consuming and makes it more difficult to scale the system to a very large number of lifts.

We believe that intelligent data analysis techniques and, in particular, deep learning, can be used to solve this problem in a more precise and adaptable way. We explain in this article how state-of-the-art deep learning architectures designed for general image classification tasks as well as dedicated domain adaptation methods can be used to improve and generalize the results obtained by the company.

2 Background on Neural Networks and Related Work

Deep learning has become synonym of learning with deep (more than 2 layers) convolutional neural networks (see [4] for some bases on neural networks). This learning technique has become tremendously popular since 2012 when a deep architecture, called AlexNet, proposed by A. Krizhevsky et al. [7] was able to win the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) by showing an outstanding improvement of the classification results, with an error rate of 15% (down from a 25% error rate for the second participant). Since then, countless different architectures have shown excellent performances in many domains such as computer vision, natural language processing or speech recognition [4].

The amount of labeled data available, the systematic use of convolutions, the better optimization techniques and the advances in manufacturing graphical processing units (GPU) have contributed to this success. However, **deep** networks were supposed to suffer from (at least) three curses. 1) The deeper the network, the more difficult it is to update the weights of the first layers (the ones closer to the input): deep networks are subject to *vanishing and exploding gradient* that leads to convergence problems. 2) The bigger the network, the more weights need to be learned. In statistical machine learning, it is well known that more complex models require more training examples to avoid overfitting phenomena and guarantee a relatively good test accuracy. 3) Neural network training aims at minimizing a loss which is a measure of the difference between the computed output of the network and the target. The computed outputs depend on (i) the inputs, (ii) all the weights of the network and (iii) the non linear activation functions that are applied at each layer of the network. The function to minimize is thus high dimensional and non-convex. As the minimization is usually achieved using stochastic gradient descent and back-propagation, it can easily get trapped in local optimum.

In this paper, we are interested in deep learning approaches developed for image classification. In such context, the use of convolutions (i.e., a reduced number and a better configuration of the weights) and the huge size of the current

image datasets ([2,8]) partially solved the second and third curses mentioned above. In the following, we will show other architectures and learning "tricks" that led to practically alleviate all the remaining problems with deep learning for image classification.

Learning tricks. The first improvement concerns the *initialization of the weights*. It was shown [1] that image classification tasks could all benefit from a pre-training step on a large dataset such as ImageNet [2]. The ImageNet dataset contains images grouped in 1000 different classes. Practically, the output of a network trained on ImageNet consists in a 1000-D vector which gives the probability for an input image to be of each of the 1000 classes. To use such a network on a new task (with much fewer and different target classes), the idea is to "cut" the last layer(s) of the pre-trained network. Only the convolutional part is kept and it's output is considered as a generic image descriptor. A new network can be build from this pre-trained generic feature extractor by training new additional "classification" layers (usually fully connected). The whole network can then be further *fine-tuned*, i.e., trained end-to-end on the new problem of interest.

The second improvement is the *batch normalization* [6]. This simple yet very effective idea consists in normalizing the activations output of each layers according to the current input batch. Its main benefit is to reduce the internal covariate shift. It corresponds to the amplification of small changes in the input distribution after each layer, and so create high perturbations in the inputs of the deepest layers.

CNN architectures. K. Simonyan et al. [9] created VGGNet 16 and VGGNet 19 composed of respectively 13 and 16 convolutional layers followed by 3 fully-connected layers. As in AlexNet, the spatial dimensions were reduced with max-pooling layers. However, each time the spatial dimensions were divided by two, the number of filters in the next layers was doubled to compensate for the information loss. With this method Simonyan et al. managed to win ILSVRC 2014 with an error rate of 7%.

Several deeper other techniques have been designed after this. For instance, He et. al. [5] introduced the concept of *residual mapping* which allowed them to create a network with 18 to 152 convolutional layers called a Residual Network (ResNet). Their architecture is divided into blocks composed of 2 or 3 convolutional layers (depending on the network total depth). At the end of a block, its input is added to the output of the layers. This sum of an identity mapping (or shortcut) with a "residual mapping" has proven effective at overcoming the vanishing gradient phenomenon during the back-propagation phase and allows to train very deep networks. An example of such blocks is shown in Figure 2. Using residual blocks, the network is learned faster and with better performance. At ILSVRC 2015, ResNet 152 showed the best performance with less than 4% of error rate. They also used *batch normalization* [6] on the first (or first two) layer(s) of each residual block.

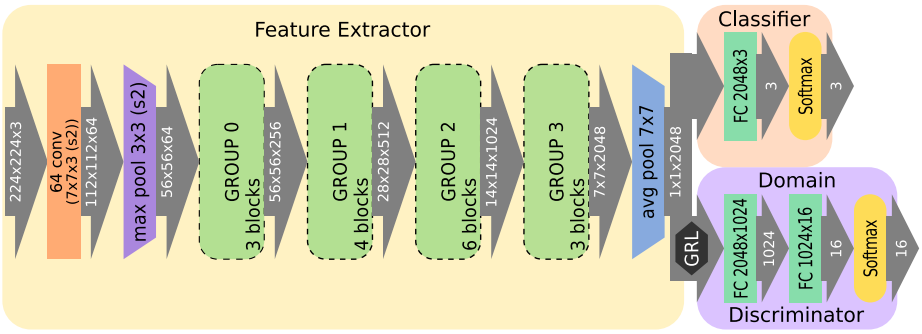

Fig. 1. The proposed ResNet architecture with domain adaptation, for a 3-class classification problem with 16 domains (chairlifts).

Domain adaptation. Some applications require to be able to learn a model on some data (e.g. images of cats) called the *source domain* and deploy it on similar but different data (images of tigers) called the *target domain*. This well-known machine learning problem is called *Domain Adaptation*. Many interesting approaches have been proposed in deep learning to tackle this problem. One of the most interesting one is from Ganin et al. [3]. The idea (as shown in Figure 1) is to train two networks that share the same first (convolutional) layers called "feature extractor". The first network is dedicated to the classification task on one particular domain. The second network aims at predicting the domain of an input example. Note that to train this second network, the examples of the target domain do not need to be labeled (and are not in [3]). The only information needed to train the second network is whether an example belongs to the source or to the target domain. The two networks are trained in an adversarial way according to the shared layers (using a mechanism called the gradient reversal on the second network optimization). As a consequence, the shared features of the networks are discriminant for the classification task as well as domain invariant.

3 Proposed Deep ResNet with Domain Adaptation

Based on the state-of-the art study presented in Section 2, we propose an image classification architecture using domain adaptation and a convolutional residual network pre-trained on ImageNet. This architecture is shown in Figure 1 and is divided into three parts: 1) a feature extractor which learns a new image representation; 2) a classifier which predicts the class of an image, and 3) a domain discriminator which ensures that the feature extractor is domain invariant, to improve classification accuracy on unseen data.

3.1 Network Architecture

The network inputs are RGB images of size 224×224 that can be viewed as three-dimensional tensors (two spatial dimension and one for the RGB channels).

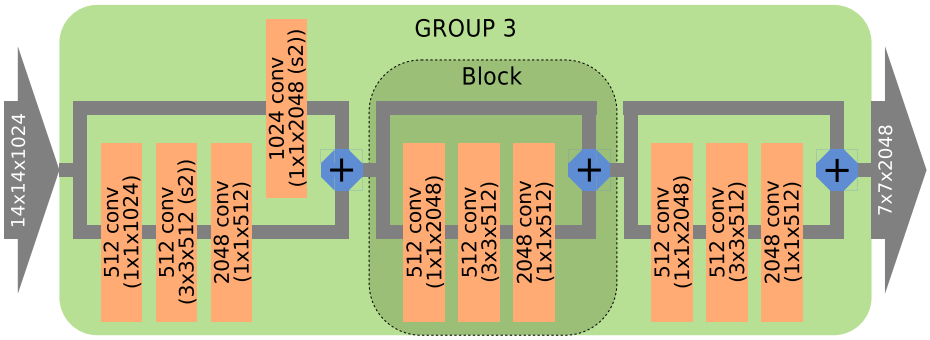

Fig. 2. Details of the last group of layers in our architecture (see Fig. 1), which is composed of three residual blocks and different convolution filters.

As illustrated in Fig. 1, we use a ResNet with 50 layers which gives a good trade-off between computational efficiency and accuracy. This network is composed of 49 convolutional layers and only one fully connected at the end. This last layer is preceded by an average pooling layer which reduces the size of the feature maps to a 1D-vector. As explained above, using our chairlift dataset, we learn a replacement for the output layer to have one output value per class (3 in our case) and then fine-tune the whole model.

In the feature extractor, each group is composed of a set of blocks. Each block is composed of three layers: a 1×1 convolution that acts as a learnable dimensionality reduction step, a 3×3 convolution that extracts some features and a 1×1 convolution that restores the dimensionality. As explained in Section 2, each block also contains an identity mapping that adds its input to its output before sending it to the next block.

The first block of each group contains a strided 3×3 convolution which reduces the spatial size by a factor two. To compute the block output with the sum operator, the input needs to have the same dimensions as the output. Therefore, a strided 1×1 convolutional layer is added to reduce the spatial size and also to match the number of feature maps which changes according to the group. In Figure 2, we show in more details the last group of layers in the feature extractor.

3.2 Objective Function and Training

Different losses are classically used in deep learning. The best known is the cross-entropy [4]. After experimenting with different losses, we decided to use the more robust multi-class hinge loss (eq. 1) commonly used in SVM:

$$loss(W, X, t) = \frac{1}{b} \sum_{i=1}^{b} \frac{1}{|C|} \sum_{c \in C \setminus \{t\}} max(0, o_c^i - o_t^i + m) \tag{1}$$

where W is the set of all parameters in the network, X is the subset of the input dataset (mini-batch) fed to the network during a training pass, d is the mini-batch size, C is the set of classes, and t is the ground-truth corresponding to each example of the mini-batch. The margin m can be seen as the minimal "distance" required between the computed probability of the prediction of the target class (o_t^i) and the others classes (o_c^i), since the loss value is equal to 0 when $o_t^i \geq o_c^i + m$.

To update the weights of the network, a stochastic gradient descent algorithm is applied on the loss function. The weight update rule is given by:

$$W(\tau + 1) = W(\tau) - \eta \cdot \frac{\partial\, loss(W(\tau), X, t))}{\partial\, W(\tau)} \tag{2}$$

where τ is the current iteration and η is *the learning rate* which needs to be tuned.

4 Experiments

The current system deployed by Bluecime (introduced in Section 1) can be decomposed into two parts: a *vehicle detector* and a *situation classifier*. The purpose of the vehicle detector is to determine the vehicle position, scale, and orientation so as to track it during each passing in front of the camera. The situation classifier detects different vehicle parts with respect to the position given by the vehicle detector. Its purpose is to detect whether the vehicle is empty or not and also to infer the position of the restraining bar, so that it can classify a frame into safe or unsafe label. A detection zone is pre-configured and at the end of it the system computes a final decision based on the series of predictions made by the classifier. Depending on the predicted label, an alarm is triggered or not.

In this workflow, the proposed network could replace the situation classifier to enhance the classification performance while keeping the very efficient vehicle detector designed by Bluecime. In the following sections, we evaluate the performance of our deep learning approach applied on images provided by the vision-based vehicle detector. We also compare those results with the ones obtained by the company with their full system.

4.1 Chairlift Dataset

Bluecime has already equipped 16 chairlifts (here named A, B, ...) from 3 different ski-resorts with the system presented in Section 1. Each time a vehicle (chair) passes in front of the camera, the system records 2 images that are approximately centered on the chair: one far from and one close to the camera. Examples of such images are given in Figure 3. A total of 50 000 such images have been recorded and manually labeled in 3 categories: *Empty*, *Safe* and *Unsafe*. In the first case (*Empty*), the vehicle does not carry any passengers. The system should not trigger an alarm in this situation. In the second case (*Safe*), the vehicle carries passengers who closed the restraining bar completely. In the last

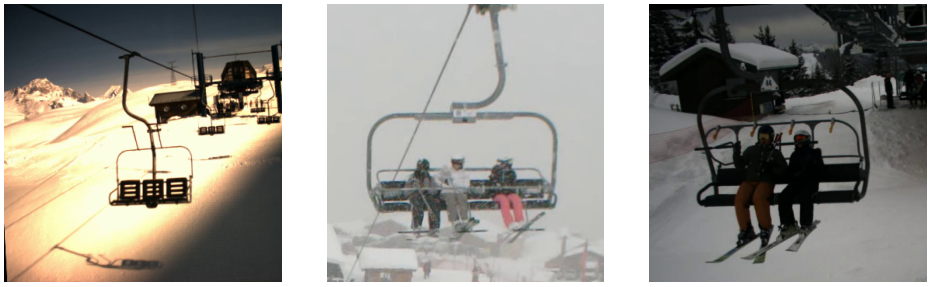

Fig. 3. Training images from three different chairlifts (labeled from left to right *Empty*, *Safe*, and *Unsafe*)

case (*Unsafe*), the vehicle carries passengers and the restraining bar is slightly to completely open.

The dataset is unbalanced with respect to the classes: around 28 000 images are labeled *Empty*, 18 000 are labeled *Safe*, and 4 000 are labeled *Unsafe*. The dataset is also unbalanced with respect to the different chairlifts (which will be called *domains* in the following): the least represented chairlift has 1 800 examples, whereas the most represented one has 6 200 examples.

As they were designed by different manufacturers at different times, all the 16 chairs in the dataset are different, though some are similar. There are even some unique cases, for instance a chairlift having a glass bubble (chairlift D) as a second protection, or another whose vehicles don't have a complete frame (see Fig. 3, left).

4.2 Evaluation Procedure

The current Bluecime system triggers an alarm when the restraining bar of the system is not in a safe position when the vehicle is not empty. They thus merge the classes *Empty* and *Safe* into a Safe category.

To study the behavior of our approach, we considered six different experimental settings:

1. **OOC** ("Only One Chairlift") 16 independent experiments are averaged: each chairlift is considered independently (as it is done by the Bluecime system), the training and the test set containing images of only one chairlift.
2. **ALL** One experiment is performed using 85% of all the available images (from all chairlifts) for training and 15% for testing.
3. **ALL DA** ("Domain Adapation") same as ALL but using the domain adaptation component described in Section 2.
4. **LOCO** ("Leave One Chairlift Out") 16 experiments are averaged: in each, all but one chairlifts (thus 15) are mixed in the training set and the images of the remaining chairlift are used as the test set.
5. **LOCO DA-** Same as LOCO but with domain adaptation using *no examples* from the target chairlift.

6. **LOCO DA** Same as LOCO but with classical domain adaptation where some *unlabeled* examples of the target chairlift are used by the domain adaptation component.

In the first setting, we can tune the hyper-parameters for each domain but training one model per chairlift can become expensive to maintain and long to configure. We expect the network to quickly overfit our data and also to be penalized by the lack of examples especially for the least represented chairlifts.

In the second and third settings, we can train our network using all the training data available. We only build one model for all the chairlifts, thus it should be less time consuming than the first setting. However, the hyper-parameters of the system are also global which may harm the final performance. These settings could be used with the current cameras installed by the company but do not evaluate the real ability of our system to work on new chairlifts.

The fourth to sixth settings really show the potential of our approach. Ideally, our method should show good enough performance on these settings to allow Bluecime to deploy their system equipped with this model on any new chairlift with no manual labeling. In these settings we expect a performance drop compared to the OOC or ALL settings because of the variability of the different domains.

To evaluate their system, the company relies on numerous measures. Among these, 4 statistical measures assess the overall performances of the system: recall, precision, F-measure, and accuracy. *Unsafe* examples are considered positive (the alarm has to be triggered), and safe examples negative (no alarm needed), thus the classes *Empty* and *Safe* are both considered negative. The *recall* gives the proportion of examples correctly detected positive among all the examples expected positive. In our case, that is the ability of the system to trigger an alarm in unsafe situations. The *precision* gives the proportion of examples correctly detected positive among all the examples detected positive. Thus it indicates the ability of the system to avoid useless alarms. The *F-measure* (harmonic mean between precision and recall) and the accuracy (percentage of correct predictions) give a more global view on the performance of the system. In the following, we report these 4 measures.

4.3 Training Details

In our experiments, we use mini-batches of 84 images (guided by GPU memory limitations), enforcing a balance over classes and domains: we randomly select a chairlift, then randomly select a class, then randomly select an example. We tune our hyper-parameters using a grid search. In all the reported experiments, the learning rate is set to 10^{-5} (with an ADAM optimizer), the hinge loss margin to 0.33 for domain adaptation component (with the reversal layer) and to 0.01 for the domain discriminator. The gradient reversal layer used for the domain adaptation induces a hyper-parameter γ set to 10 as defined by Ganin et al. [3].

Our dataset is composed of images of size 237×237. When an image is loaded, we randomly select a 224×224 crop in the image. To maximally exploit

Experiment	F-measure	Precision	Recall	Accuracy
Bluecime	84.72	87.88	81.78	97.40
OOC	**91.70**	**94.64**	**88.94**	**98.68**
ALL	89.47	93.49	85.78	98.33
ALL DA	**89.71**	**93.88**	**85.89**	**98.36**
LOCO	76.23	76.71	75.75	96.07
LOCO DA-	72.36	70.84	73.93	95.30
LOCO DA	**84.24**	**82.76**	**85,77**	**97.33**

Table 1. Performance results of our deep learning system for the three main group of settings (OOC, ALL, LOCO) compared to the hand-tuned existing system of the Bluecime company.

the small number of examples available in the positive class, we decided to not use a validation and to stop the learning process after a constant (high enough) number of iterations of 12 000. Training our model for 12 000 iterations takes about 4 to 5 hours. The classification of one image is done in approximately 15ms, so our method could be used in real time which is a necessary condition imposed by the application.

4.4 Experimental Results

In Table 1, we present the results on all the datasets for all experimental settings. The first line of the table gives the performance of the current hand-tuned method developed by Bluecime. In the OOC setting (the closest to what the company currently do), the proposed network brings a 7 point improvement in precision, recall and F-measure and +1pt in accuracy. This validates the relevance for the company of the proposed method based on machine learning. With the ALL setting, a 5pts gain of F-measure and 1pt gain of accuracy are observed. While the gain is smaller than with OOC, the *one-model-fits-all* nature of ALL makes it a very attractive solution from an industrial standpoint. Adding domain adaptation (ALL DA) has a very limited impact which can be explained by the fact that all the domains are already in the training set.

As expected, during the LOCO experiment, performance losses occur for all the measures, with -8pts of F-measure and -1pts of accuracy. In this setting, performances are too low for industrial deployment. However, adding domain adaptation (LOCO DA) bring back the performances to a level comparable with the Bluecime system, with -0.5pts of F-measure and a similar accuracy. These results emphasize that domain adaptation is meaningful and allows to create models that have competitive results even on a chairlift with non-labeled examples.

In the case when we don't have any example of one chairlift (e.g., launching the system immediately after installation) adding domain adaptation (LOCO DA-) causes a 4pts loss of F-measure (compared to plain LOCO). This behavior

Chairlift	Size	Experiment	F-measure	Precision	Recall	Accuracy
A	2842	OOC	95.68	97.08	94.33	97.06
		LOCO DA	92.98	93.56	92.40	95.21
B	6306	OOC	88.89	91.67	86.27	98.83
		LOCO DA	72.19	66.50	78.95	96.70
C	1822	OOC	76.92	78.95	75.00	96.79
		LOCO DA	77.78	83.05	73.13	96.93
D	2736	OOC	85.71	100.00	75.00	99.76
		LOCO DA	82.76	85.71	80.00	99.63
Overall Average (Table 1)		OOC	91.70	94.64	88.94	98.68
		LOCO DA	84.24	82.76	85,77	97.33

Table 2. Comparison of OOC and LOCO experiments on 4 specific chairlifts

may be unexpected but it has an intuitive explanation: domain adaptation encourages the learning of features that are domain-invariant but also specializes these features for the domains it has seen, causing a detrimental effect on an unseen chairlifts. These results show that in a LOCO setting, domain adaption should be used only if we can retrain the model using new unlabeled images, and not if the goal is to produce an off-the-shelf model.

In the Table 2, we focus on 4 different chairlifts in order to illustrate how performances vary in function of the specificity of each chair model.

Chairlift A – In both settings, results are better that the average with +4pts (OOC) and +9pts (LOCO) of F-measure. This gain in performance is possible, for OOC, thanks to a good balance in the classes of the images of this chairlift, and for LOCO, thanks to the chairlift configuration similar to several other ones.

Chairlift B – OOC has slightly lower performance with -3pts of F-measure but +0.5pts of accuracy. This is explained by the over-representation of the *Safe* class (4 600 images versus 340 for *Unsafe* class). On the LOCO setting, we observe a considerable performance drop with -12pts of F-measure. These poor performances are mostly due to the sun influence, indeed, different shadows are cast depending of the time of the day. And above all a high number of images present flares decreasing the image quality.

Chairlift C – OOC has a F-measure of 15pts below the overall result. It is even outperformed by LOCO (+1pts of F-measure and +0.1 of accuracy). Chairlift C being one of the domains having the fewest number of examples (around 1 800), in the OOC setting, this lack of data is even shows up in the statistical measures computed. In the LOCO experiment, the poor performances are mostly due to the shadow cast by the chairlift tower in the afternoon.

Chairlift D – This is the only chairlift with a glass bubble, and also the only one which has the restraining bar (and also the bubble) closed on *Empty* images. This implies that, even if its data are unbalanced (2 000 *Empty*, 670 *Safe*, and 30 *Unsafe*), on the OOC setting, the network has perfect precision and retrieve

3 of the 4 positives examples, as only 15% of the 30 *Unsafe* are used for testing. The F-measure is 6pts lower than the average result, yet it is highly impacted by this lack of positive testing examples. In LOCO, we could have expected an important performance drop considering the chairlift unique features (the bubble), but when the restraining bar is open, the bubble is open too, and so the configuration in that case is not dramatically different to other chairlifts. This explains why the loss of F-measure is limited to 2pts.

These results underline the different factors limiting the performance of the proposed system. When training and testing on the same domain we must have enough data to perform well. When testing on a new domain, if this domain is very different from the others, we can expect lower results as the domain adaptation won't be able to bridge the domain gap.

5 Conclusion

We have presented an end-to-end deep learning system that can improve the security on chairlifts using visual information. Leveraging domain adaptation techniques, the model is able to achieve competitive performance even with (new) chairlifts for which no labeled data is available. Overall, this ability to generalize, its competitive accuracy and its operation in real-time make this system well suited for industrial deployment.

Acknowledgment

The authors acknowledge the support from the *SoLStiCe* project `ANR-13-BS02-0002-01`.

References

1. Chatfield, K., Simonyan, K., Vedaldi, A., Zisserman, A.: Return of the devil in the details: Delving deep into convolutional nets. In: BMVC (2014) 3
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009) 3
3. Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V.: Domain-adversarial training of neural networks. JMLR (2016) 4, 8
4. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org 2, 5
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Computer Vision and Pattern Recognition (CVPR) (2016) 3
6. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: ICML. pp. 448–456 (2015) 3
7. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012) 2
8. Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: 13th European Conference in Computer Vision - ECCV (2014) 3
9. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014) 3