Vihlman, Mikko; Kulovesi, Jakke; Visala, Arto

Tree Log Identity Matching Using Convolutional Correlation Networks

Published in:
2019 Digital Image Computing: Techniques and Applications (DICTA)

DOI:
10.1109/DICTA47822.2019.8945865

Published: 01/12/2019

Document Version
Peer-reviewed accepted author manuscript, also known as Final accepted manuscript or Post-print

Please cite the original version:
Vihlman, M., Kulovesi, J., & Visala, A. (2019). Tree Log Identity Matching Using Convolutional Correlation Networks. In *2019 Digital Image Computing: Techniques and Applications (DICTA)* IEEE. https://doi.org/10.1109/DICTA47822.2019.8945865

Tree Log Identity Matching Using Convolutional Correlation Networks

Mikko Vihlman
Electrical Engineering and Automation
Aalto University
Espoo, Finland
Email: mikko.vihlman@aalto.fi

Jakke Kulovesi
Electrical Engineering and Automation
Aalto University
Espoo, Finland
Email: jakke.kulovesi@aalto.fi

Arto Visala
Electrical Engineering and Automation
Aalto University
Espoo, Finland
Email: arto.visala@aalto.fi

***Abstract*—Log identification is an important task in silviculture and forestry. It involves matching tree logs with each other and telling which of the known individuals a given specimen is. Forest harvesters can image the logs and assess their quality while cutting trees in the forest. Identification allows each log to be traced back to the location it was grown in and efficiently choosing logs of specific quality in the sawmill. In this paper, a deep two-stream convolutional neural network is used to measure the likelihood that a pair of images represents the same part of a log. The similarity between the images is assessed based on the cross-correlation of the convolutional feature maps at one or more levels of the network. The performance of the network is evaluated with two large datasets, containing either spruce or pine logs. The best architecture identifies correctly 99% of the test logs in the spruce dataset and 97% of the test logs in the pine dataset. The results show that the proposed model performs very well in relatively good conditions. The analysis forms a basis for future attempts to utilize deep networks for log identification in challenging real-world forestry applications.**

I. Introduction

Tree log identification aims to recognize the identity of specific known individual logs. Given information or measurements of two logs, the goal of identification is to tell whether the logs are exactly the same individual. This allows creating a database of known logs that can be later matched with new examples. Log identification is distinct from but closely related to applications such as log detection and species recognition that aim to automatically detect logs and classify their species. Identification is more complex as it requires distinguishing between very similar specimen of a given species.

Log identification has important applications in forestry. Our ultimate goal is to create a database of tree logs by imaging the trunks from multiple angles simultaneously to harvesting and cutting the trees into logs. Ideally, the harvester can image the logs and assess their quality (e.g. size, knots) without any delay in the cutting process. Then, after transporting the logs to a sawmill for additional processing, the goal is to match each log with the database. Identification enables tracing a given log back to the location where it was grown in and picking up stored logs of certain quality without repeating the quality assessment. Knowing the origin, quality and destination of each individual log increases the efficiency of sawmills. Thus, log identification can be an important tool to improve planning, efficiency and traceability of forestry.

Trees can be located using GPS; however, GPS is often inaccurate or unreliable in dense forests, and it cannot be used for log identification since logs are typically stored in stacks. Logs can be identified simply by marking each log separately, e.g. with paint or an RFID tag. However, physically marking the logs and reading the markings can be inefficient, and the markings can be cumbersome to use prior to cutting the trees. It is desirable to develop methods that are applicable more generally in silviculture and forestry. Our approach is to identify logs by matching images taken of the trunk. Since the trunk is the most stable visible part of a tree, present both before and after cutting, the developed methods can be useful more generally, throughout the process of growing, tending, monitoring, harvesting and processing the harvested trees.

To our knowledge, this is the first paper to study the given application, using machine vision to match tree trunks. Thus, to provide a proof of concept, we slightly simplify the setting; the database and example images are taken at the same time using a stereo camera system while the logs are moving on a line in a sawmill. Fig. 1 shows some examples. Identification is done based on a single frame. It is straightforward to extend our model to utilize multiple frames to improve the accuracy, although our results show that a single frame is typically enough to reliably find the matching log.

In this paper, logs are identified using a deep two-stream convolutional network. Deep networks have become common e.g. in image recognition [1], object detection [2], semantic segmentation [3] and action recognition [4]. Our network measures the likelihood that two frames represent the same part of a log. The similarity is measured using cross-correlation layers that explicitly compare the spatial convolutional features of the frames, at one or more levels of the network. A given example image is compared to each image of the database to find the most likely match. The paper is structured as follows. Section II briefly summarizes the related literature. Sections III and IV present the methodology and results. The main results are discussed in Section V. Section VI concludes.

Strategic Research Council at the Academy of Finland is acknowledged for financial support for project "Competence Based Growth Through Integrated Disruptive Technologies of 3D Digitalization, Robotics, Geospatial Information and Image Processing/Computing - Point Cloud Ecosystem (293389 / 314312)".

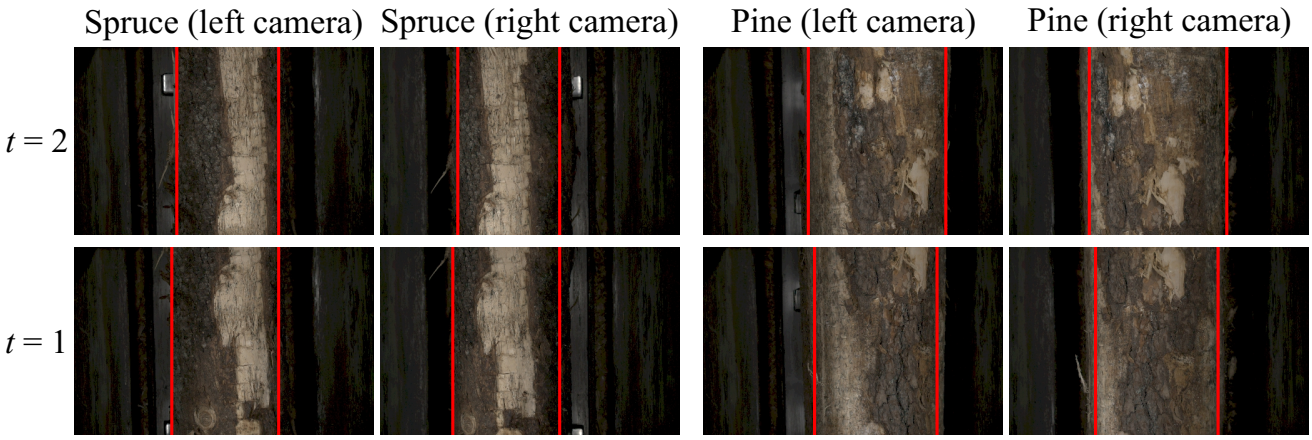

Fig. 1. Examples of spruce and pine logs. Each log is imaged as a series of stereo camera images. Each row of the figure shows one moment of time, the left half for spruce, the right half for pine. The database of known logs contains all images of all logs taken by the right camera. For a given example frame imaged by the left camera, the goal is to search the database and find the matching frame taken by the right camera at the same time. Matching is done using cropped frames that exclude most of the surroundings of the logs. The cropping window (red vertical lines) is detected automatically for each pair of frames.

II. Related Work

This paper uses a two-stream network that processes a pair of frames with identical branches of convolutional layers. Two-stream structures have been used in several applications, e.g. in signature verification [5], face recognition [6] and action recognition [4]. They are also common in visual tracking (e.g. [7]–[10]). Our application is closely related to tracking due to measuring the similarity between frames, but there are also notable differences. First, in our context, the displacement of the given part of a log can be relatively large from one image to another, whereas tracking often considers videos of a high framerate. Second, in log identification, there is no exact object and relatively large part of the trunk seen in one image may not be visible in another. Third, we compare two static frames, whereas visual tracking can utilize information over time.

The proposed model estimates the cross-correlation of deep convolutional feature maps. Similar approach is used e.g. by FlowNet [11] and SiamFC [12]. FlowNet estimates the optical flow between two frames using the correlation of deep features. SiamFC uses template matching for visual tracking by calculating the cross-correlation of convolutional features. It is an example of similarity learning, which has been studied also in [13] and [14]. Our network is highly similar in structure to SiamFC, albeit using the correlation map for classification instead of estimating the displacement. Contrary to SiamFC and Flownet, our model can utilize correlations from multiple levels of the network. Another fundamental difference is that our model needs to provide useful information also when the given pair of frames represent different objects (different logs).

Since we currently focus on matching images taken by two cameras at the same time, identification could be based on stereo matching. For instance, [15] computes the inner product of two convolutional feature maps for stereo matching of image patches, and [16] concatenates convolutional features of two branches to extract depth information from a pair of images. However, our ultimate goal is to match images taken separately in different locations. Developing methods that are not based too much on stereo matching allows using the methods for log identification in more general conditions.

Log identification is also related to person reidentification (re-ID), e.g. matching images of pedestrians from multiple cameras (see e.g. [17] or [18]). Although people and tree logs look fundamentally different, similar methods could be useful in both applications. Identification in re-ID is often done either using binary classification (match or not) or by assigning each person with a unique ID predicted by the network. One recent trend [17] is to merge the two approaches and train the network using two loss functions. We currently use only binary classification for log matching; using the multi-class approach or other re-ID methods could improve the performance.

Relatively few papers on computer vision focus specifically on forestry and even fewer on log identification; the relevance of methods evaluated in more general context might be low. Thus, it can be useful to look at related forestry applications like tree detection and species recognition. For instance, [19] uses simple traditional machine vision methods for detecting trees and classifying their species. [20] reviews literature on multiple methods for classifying tree species based on leaves or barks of trees and discusses approaches to fuse the two sources of information. [21] uses local binary patterns [22] to analyze the texture of barks for species recognition, and [23] uses convolutional networks for leaf-based species recognition.

III. Methods

A. Data and Preprocessing

Deep networks need large datasets for learning and testing. In real-world forestry applications, however, collecting realistic large datasets is often difficult. Imaging trees while harvesting and processing logs is time consuming, inefficient and expensive. Thus, this paper focuses on a simplified case by imaging the database and test images at the same time from one side of the logs using a fixed stereo camera system.

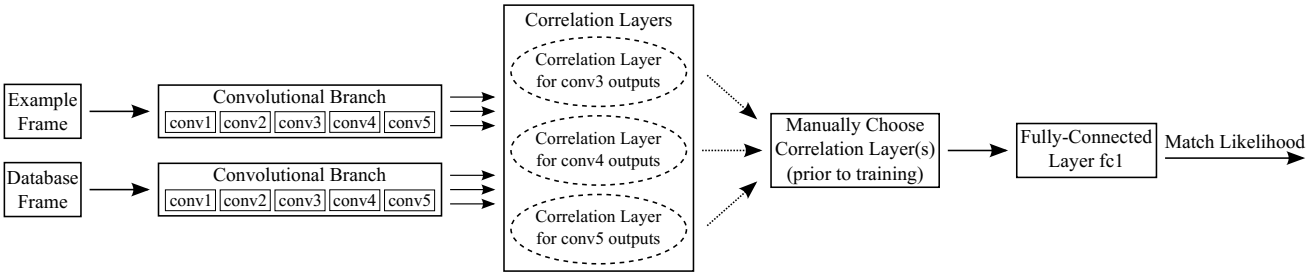

Fig. 2. Network architecture for matching a given pair of frames. The network gets two input frames (example, database) that are cropped automatically to remove the surroundings of the tree log. The network estimates how likely the cropped input frames are a match (represent the same part of a tree log). The inputs are processed by two identical branches of five convolutional layers with shared parameters. The outputs of the third, fourth and fifth convolutional layers are fed pair-wise into cross-correlation layers. The architecture is fixed prior to training by choosing which correlation layers to include in the model, to potentially compute correlations only at some levels of the network. Matching is done using a fully-connected layer on top of the correlation layers.

While imaging, the logs move along a line in a sawmill at a fairly constant speed. The framerate is relatively low; two consecutive frames taken by a given camera overlap by about one third. The images taken by the first (left) camera are used as example images that should be matched with the database. The database contains the images taken by the second (right) camera. Using stereo pairs limits the ecological validity of the results; however, we consider the approach sufficient for a proof of concept, noting that the current context has not been extensively studied in prior literature.

Images are preprocessed prior to the analysis. First of all, the first and last image of each log that contain only a short piece of the log are discarded. Second, the images are cropped to remove the surroundings of the logs. A few of spruce images are ignored automatically due to a poor cropping result. Third, the cropped images are resized to the input size of the network (227×227 pixels). Finally, the average RGB color, computed for the images of the Imagenet Recognition Challenge [24], is subtracted from the frames. For simplicity, the cropping windows are detected for the first camera, and each window is moved to the second camera using a fixed displacement (the cameras are fixed to each other during imaging).

The cropping windows are detected automatically using the MATLAB computing environment. Feature vectors containing about 1000 KAZE features [25] are extracted in informative locations around the image. KAZE features aim to be invariant to differences e.g. in lighting and viewing angle. Features computed for very bright or colorless regions are discarded as they tend to represent the surroundings of the logs. The computed features are divided vertically to five groups. The edges of the trunk are found as two vertical lines such that there are features from each group inside the two lines. In this paper, KAZE features are used only to crop the input frames; they are not used by the network for matching.

Two datasets are collected, containing either spruce or pine logs. After preprocessing, the spruce dataset contains 4062 pairs of images of 214 spruce logs, and the pine dataset contains 8224 pairs of images of 506 pine logs. In both datasets, each log is represented by about 15–20 pairs of frames. The two datasets are processed individually; the models are trained and evaluated separately for each species.

TABLE I
LAYER DETAILS OF PROPOSED ARCHITECTURE.

Layer	Parameters	Output	Notes
Inputs	–	3x227x227	RGB crops (log images)
conv1	3x11x11	96x55x55	stride 4, ReLU
Max Pooling	–	96x27x27	kernel 3x3, stride 2
Local Norm.	–	96x27x27	
conv2	96x5x5	256x27x27	stride 1, ReLU
Max Pooling	–	256x13x13	kernel 3x3, stride 2
Local Norm.	–	256x13x13	
conv3	256x3x3	384x13x13	stride 1, ReLU
conv4	384x3x3	384x13x13	stride 1, ReLU
conv5	384x3x3	256x13x13	stride 1
Correlations (1–3 layers)	–	Output has a single channel; spatial shape from the conv inputs	
fc1	n x 2	2	n: number of elements in correlation outputs

B. Network Architecture

Fig. 2 shows the studied architecture. The network estimates the likelihood that two input frames represent the same part of a log by estimating the cross-correlation of deep feature maps. More specifically, the network gets two cropped inputs; a new example and a frame from the database of known logs. Identical branches of convolutional layers compute features for both images. Cross-correlation layers compare the features of the branches, at specific convolutional levels. Finally, the correlation outputs are concatenated and fed into a single fully-connected layer to estimate how likely the input images are a match. This process is repeated for each image of the database to find the most likely match for the given example image.

The convolutional branches are identical; they contain five convolutional layers that have the same structure as the first five layers of CaffeNet (see Table I and [26]). CaffeNet is analogous to AlexNet [1] that reached seminal performance in the ImageNet Recognition Challenge [24] in 2012. CaffeNet is simple and traditional and sufficient for our dataset. Modern architectures like ResNet [27] could improve the performance.

As shown in Fig. 2, the user chooses which convolutional levels have a correlation layer. This paper studies four architectures. The first three architectures have a single correlation layer, either for conv3, conv4 or conv5 outputs. The fourth architecture has three correlation layers; the first compares the outputs of conv3, the second the outputs of conv4, and the third the outputs of conv5. The inputs of the correlation layers are taken prior to applying ReLUs, pooling and normalization.

A correlation layer gets two convolutional feature maps of shape $[n_\mathrm{f} \times n_\mathrm{h} \times n_\mathrm{w}]$, where n_f is the number of features (channels) and n_h and n_w are the spatial height and width. The computation is similar to convolution. First, the database map is centered at the corner of the example map, and the inner product (the correlation) of the two maps is collected. Second, the database map is moved horizontally and vertically, pixel by pixel, to collect the correlation in each location of the example map. The correlation output has a single channel and the same spatial shape as the inputs of the layer. The feature maps are padded with zeros when necessary. The output is divided by the number of elements in one feature map and by 255 (maximum RGB value) to reduce the absolute value.

Fig. 3 shows how cross-correlation can be used to match a pair of convolutional feature maps. The correlation output is a map; moving along vector d from the center of the map to a specific location tells how similar the feature maps are when the database map is shifted along d. When the feature maps match (left side of Fig. 3), the output correlation map typically contains a bright spot indicating the most likely displacement between the input images. When the inputs do not match (right side of Fig. 3), the correlation map is darker and more diffuse.

The output of the network is produced by a fully-connected layer with two outputs (match, no match) that can be used to measure how likely the given pair of cropped frames are a match. When the network has more than one correlation layer, the outputs of the correlation layers are concatenated into a vector prior to feeding them to the fully-connected layer.

The proposed architecture has some limitations. First, the correlation map ignores the scale of the input frames and is computed using the full extent of the convolutional feature maps instead of smaller patches. This limits the ability to match images with less overlap or taken from different distances from the trunk. Second, a single fully-connected layer has a limited capacity to learn to recognize matches that can appear anywhere in the correlation map. However, the architecture is sufficient for learning convolutional parameters. Also, as long as the database is imaged fairly densely, it should contain at least a couple of frames with a relatively fixed displacement from the given example frame.

C. Training and Evaluation

In this paper, the proposed model is trained and evaluated separately for two datasets, each containing either spruce or pine logs. Both datasets are split to separate parts for training (50%), validation (30%) and testing (20%). All frames of a given log are placed in the same split.

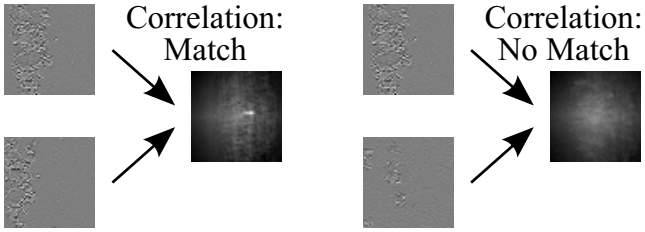

Fig. 3. Illustration of using the cross-correlation of the convolutional features for matching. On the left, the convolutional maps (only one channel shown) are very similar, and the single-channel correlation map contains a bright spot that shows high similarity for a specific displacement between the convolutional maps. On the right, the convolutional maps are not similar enough, and the correlation output is diffuse and flat, suggesting that the input frames represent different logs or different parts of a log.

During training, a softmax layer converts the two outputs of the network into a match probability. Training is done in mini-batches using Stochastic Gradient Descent (SGD) guided by the multinomial logistic loss between the produced and real probabilities. Each mini-batch contains a positive (true) pair and a negative (false) pair for random 64 training logs. For a positive pair, the frames are taken by the two cameras at the same time. For a negative pair, the frames are of different logs. The target probability is 1 for positive pairs (match) and 0 for negative pairs (no match). The network is trained for 50,000 iterations which is enough for the loss function to converge, regardless of the studied learning rates and architectures (see below).

The fully-connected layer is initialized with small random weight parameters and small biases and updated using a base learning rate of 1e-3. The convolutional layers are either initialized and learned like the fully-connected layer or pretrained for the ImageNet Recognition Challenge [24]. While using the pretrained parameters, the convolutional layers are updated like the fully-connected layer, or the base learning rate can be reduced to 0 or 1e-4 to either fix or finetune the parameters.

The learned representations are validated in two steps to choose the best settings for initialization and learning speed (Section IV-A) and the best architecture (Section IV-B). As explained in Section III-B, this paper considers four architectures that have either a single correlation layer (for conv3, conv4 *or* conv5 outputs) or three correlation layers (for conv3, conv4 *and* conv5 outputs). Processing the pine dataset is more time consuming due to the larger number of logs. Thus, for simplicity, the settings and architecture are chosen only based on the spruce dataset. In Section IV-C, the final evaluation is done by applying the same chosen settings and architecture for both datasets.

While evaluating the training, validation or testing performance, there is always exactly one correct candidate for each example frame; the same location imaged at the same time by the database camera. The database of incorrect candidates is formed by the frames of all other logs of the whole dataset (of the given species), except that the test logs are ignored until the final evaluation of the chosen architecture. The tests are implemented using the Caffe deep learning framework [26].

TABLE II
EFFECT OF LEARNING MODE ON MATCHING ACCURACY.

Correlation Inputs	Learning Mode	Training	Validation
conv5	Fixed	54.1%	56.6%
conv5	Finetune	99.9%	99.5%
conv5	Normal Speed	99.8%	99.1%
No. of Spruce pairs (logs)		2062 (108)	1212 (64)
False candidates from all training and validation logs.			

TABLE III
EFFECT OF ARCHITECTURE ON MATCHING ACCURACY.

Correlation Inputs	Training	Validation
conv3	99.7%	99.3%
conv4	99.8%	99.2%
conv5	99.9%	99.5%
conv3 & conv4 & conv5	99.6%	99.4%
No. of Spruce pairs (logs)	2062 (108)	1212 (64)
False candidates from all training and validation logs.		

IV. RESULTS

A. Transfer Learning

This section presents how the initialization of the parameters and the learning speed affect the performance of the candidate network that has a single correlation layer on top of the conv5 outputs. It was found that a randomly initialized network does not learn anything; the performance stays random regardless of the learning rate. Thus, all networks in this paper are initialized with the convolutional parameters pretrained for the ImageNet Recognition Challenge [24].

Table II shows the correct matching rates (proportion of log images matched correctly) for the spruce dataset using three learning modes; the convolutional parameters are either held fixed, finetuned slowly or learned normally like the parameters of the fully-connected layer. Tuning the convolutional parameters seems necessary; the architecture with fixed pretrained parameters identifies correctly only about 55% of the log images. The difference between finetuning or learning at the normal learning rate is small. Finetuning gives slightly better matching rate for the validation logs; thus, the remainder of this paper trains the networks using finetuning.

B. Architecture

This section presents how the architecture affects the performance of the proposed model in the spruce dataset. The candidate architectures have either a single correlation layer on top of a specific convolutional layer or three correlation layers on top of the last three convolutional layers. All architectures are trained by finetuning the pretrained convolutional parameters slowly while learning the parameters of the fully-connected layer at the normal speed.

Table III shows the correct matching rates (proportion of log images matched correctly) for the spruce dataset using the four candidate architectures. Overall, the architectures perform very similarly. The architecture that has a single correlation layer on top of the fifth layers (conv5) of the two convolutional branches is slightly the best both in training and validation; thus, the final evaluation is done only using that architecture.

C. Final Evaluation

This section presents the final performance using the settings and architecture chosen in the previous subsections. The network has a single correlation layer on top of the conv5 outputs, and the pretrained convolutional layers are finetuned slowly while learning the classification layer at the normal speed. Table IV shows the final performance (proportion of log images matched correctly) when the model is trained separately for the spruce and pine datasets. There are some signs of overfitting as the performance is slightly worse for the validation and testing data. However, the accuracy is very good even for the test logs that were not used for training, validation or developing the methodology. The model performs slightly worse for pine logs than for spruce logs.

TABLE IV
FINAL MATCHING ACCURACY.

Dataset	Correlation Inputs	Training	Validation	Testing
Spruce	conv5	99.9%	99.5%	98.8%
Pine	conv5	98.5%	97.7%	97.1%
No. of Spruce pairs (logs)		2062 (108)	1212 (64)	788 (42)
No. of Pine pairs (logs)		4129 (253)	2468 (152)	1627 (101)
False candidates from all logs, separately for each dataset.				

Looking more carefully at the identification process for each image reveals cases where the network is i) confident and correct, ii) less confident but correct, iii) confident but incorrect, and iv) less confident and incorrect. Fig. 4 shows some examples of these cases. Each row in the figure shows first an example frame that should be identified and then four database frames that the trained model thinks are the most likely matches. The probability that a given database frame is a match for the given example frame is computed and displayed separately for each database frame. The ground truth match is shown in red rectangle if it is in the top four.

The first row of Fig. 4 shows a successfully identified spruce log. The network is almost certain that the best match is correct and that the other database frames are incorrect. The top four database frames are all very similar to the example, containing a brighter vertical region on the right side of the log. The second row of Fig. 4 shows a failed matching of a spruce log. In this case, the automatic edge detection is inaccurate for the example crop, and the ground truth match is not in the top four. In this case, the network is less confident; it gives the most likely match a probability of less than 60%.

The third row of Fig. 4 shows a successfully identified pine log. This case is fairly difficult since the most distinguishable bright regions in the lower part of the image are visibly different in the example frame and its ground truth match. The difficulty is reflected by a relatively low matching probability of 72%. The fourth row of Fig. 4 shows a failed matching of a pine log. This case is interesting in at least two ways. First, the automatic edge detection for the example image is inaccurate; the whole breadth of the log is not inside the cropping window.

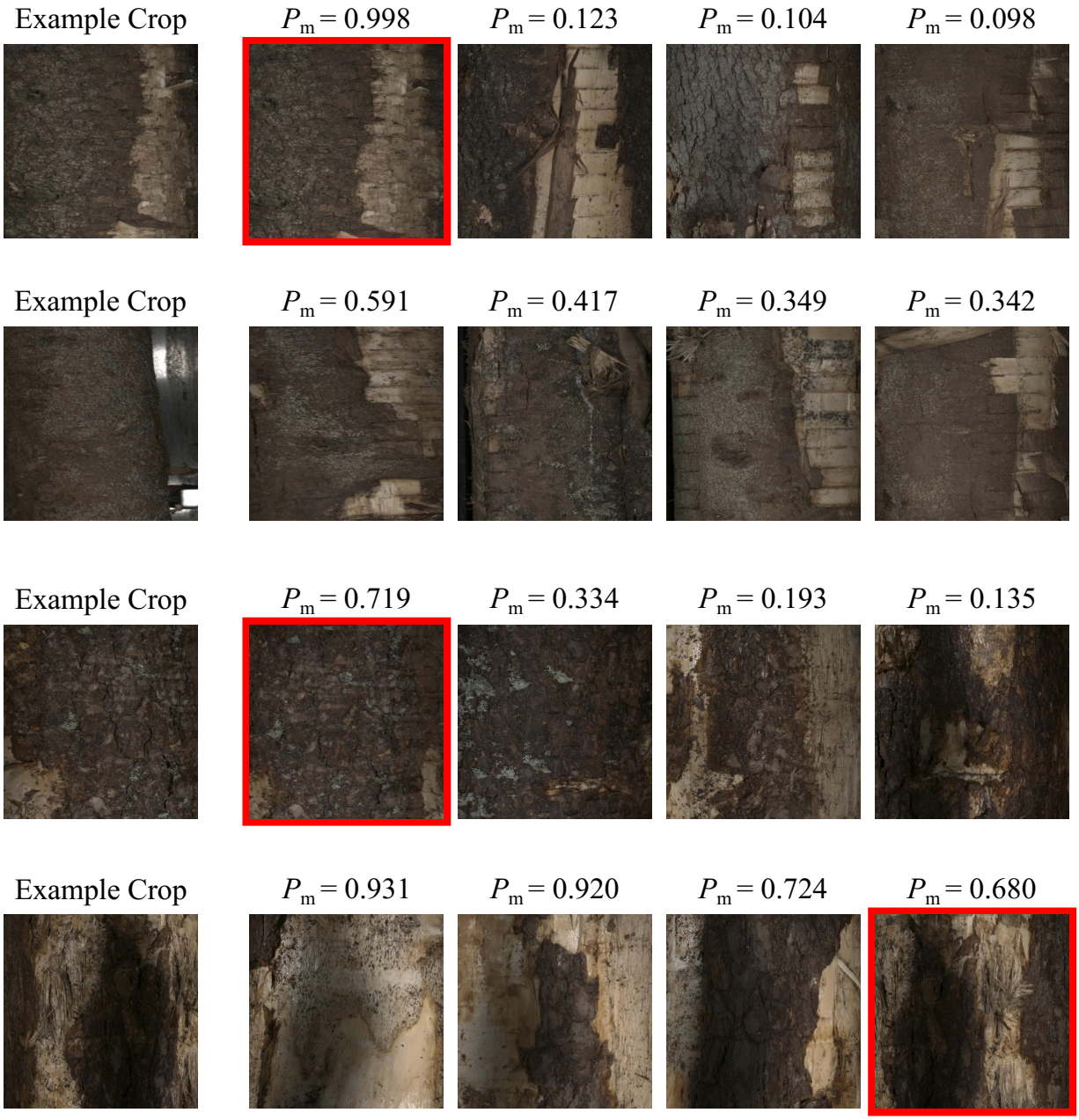

Fig. 4. Examples of the identification process. Each row shows one example frame followed by its top four most likely matches from the database. The probabilities P_m measure how likely a given database frame is a match. Red rectangle shows the ground truth match if it is in the top four.

In effect, the cropped example and its ground truth overlap rather little, making matching very difficult (matching is done for the full crop). Also, the fourth example contains a dark pattern, likely a shadow of an object in the mill while imaging the log. The top four matches all show the same shadow.

V. Discussion

The final test results using separate datasets show that the proposed model works very well in practice. For both spruce and pine, the model finds the correct match for most log images, from databases of thousands of images. The identification accuracy is slightly better for spruce than pine; however, the model works almost perfectly for both species.

The results show that our methods effectively transfer previously learned representations to the context of identifying individual tree logs. Our log datasets are not extensive enough for learning informative features without pretraining. On the other hand, directly using parameters pretrained for large-scale image recognition provides only moderate performance (accuracy of about 55%). Reaching good performance requires additional finetuning to effectively transfer the pretrained representations to the current context. The learning rate used to update the convolutional parameters does not notably affect

the accuracy; the pretrained parameters might serve mainly as an effective initialization. However, the learning rates might have been sufficiently small so that the network still utilizes features relatively similar to the pretrained network.

Multiple architectures were compared using the training and validation datasets of the spruce logs. The results show that the architecture has relatively little effect on the identification accuracy. In the chosen architecture, correlation is computed between the outputs of the fifth layers of two identical convolutional branches. Similar accuracy is reached by taking the correlation between the third or fourth layers or at multiple levels of the branches. This can be explained in at least two ways. First, the last three convolutional layers might learn representations that are about equally useful for log identification. Second, the higher layers might learn more informative features, but the benefit is reduced due to the smaller spatial shape of the correlation map. The shape also affects the number of parameters in the classification layer.

The relatively small effect of the architecture has important implications for the real-world forestry applications. Sawmills process large numbers of logs at a high pace and have limited computational capacity. The user can choose the architecture most efficient for the target hardware and software platforms. For instance, computing the correlations of only the conv3 outputs allows removing the fourth and fifth convolutional layers from the network. On the other hand, the conv4 and conv5 outputs have a smaller spatial shape, requiring less storage space for the convolutional features precomputed for the database. If matching correlations takes most of the time, it might be more efficient to transfer the outputs of smaller shape to an efficient remote cloud service.

The proposed model performs slightly worse for pine than spruce. In our dataset, identification might be more difficult for pine logs since their bark is often partly or mostly missing. The surface might be more distinctive when the bark is unbroken, although the lack of bark does not seem to notably decrease the performance. Running the validation process also for the pine dataset might improve the accuracy.

The pine dataset contains about twice as many incorrect candidates for matching as the spruce dataset (8000 versus 4000), which can make it more difficult to find the single correct match. The identification accuracy tends to decrease in the size of the database; however, this effect seems to weaken when there are at least about 1000 incorrect candidates. This finding is very relevant since practical databases are likely to contain at least about ten times as many frames as the one used in this paper. Real databases must have images from all sides of the logs and the images must be flipped in the vertical direction to account for the different orientations of the logs.

There are several ways to potentially improve the accuracy of the proposed model. Our preliminary tests suggest that identifying logs using multiple frames improves the identification accuracy, potentially so much that finetuning the convolutional parameters becomes unnecessary. However, at least for the current datasets, a single frame is enough to identify the logs very reliably. This paper used downsampled frames so small that many frames had to be increased in size after cropping the frames to match the input size of the network. The quality of the input frames can be increased without changing the input size of the network since our cameras take high-resolution images. High-resolution images would also allow increasing the input size, although this decreases efficiency and might require repeating the pretraining process. Finally, although the automatic edge detection for cropping the frames prior to running the network works very well, our results show that inaccurate edges sometimes lead to identification errors.

The way the network estimates the correlation map and converts it into the match likelihood is subject to limitations. The correlation map is computed using the full extent of the convolutional feature maps instead of comparing smaller patches, and the potential differences in the scales of the input frames are ignored. Thus, the resulting map is likely to be inaccurate when the input frames overlap relatively little or are taken at different distances from the trunk. Considering classification, since the frames of positive pairs (true matches) were imaged at the same time by a stereo camera system, the fully-connected layer learns that matching frames must have high responses in a specific location of the correlation map. Thus, the trained network performs poorly when there is a larger vertical shift between matching images or when the example frame is horizontally from the other side of the correct database frame. Our datasets ignore these cases; thus, the implicit assumption is that the database is imaged densely.

More realistic datasets are needed for testing, e.g. creating the database while harvesting and matching later in a sawmill. Our dataset contains some variability between the examples and ground truth matches due to the difference in the viewing angles of the cameras and the inaccuracies of the automatic cropping method, but the differences tend to be small. More challenging datasets might need more modern and powerful architectures (e.g. ResNet [27] instead of AlexNet [1]). It is also important to estimate the number of image pairs needed for training and to compare the performance of convolutional neural networks to other common methods like Local Binary Patters [22] or Histograms of Oriented Gradients [28].

It is necessary to study the robustness to changes in logs after creating the database. Moisture, lighting, living organisms and mechanical operations can damage or otherwise change the visual appearance of the logs. Using multiple images from the log could increase reliability when the changes are local. One practical limitation is that some processing phases in the real forestry processes might involve removing the bark entirely, making the previously imaged database unusable. In that case, a given log must be identified right before the bark is removed, and the database must be updated to maintain traceability. This can be cumbersome since the log must be imaged from all sides after removing the bark.

The implementation is currently not efficient enough to use in sawmills. Even on a relatively efficient GPU, matching one example frame with a database of 4000 images takes about 10 seconds. About one third could be avoided by precomputing the convolutional features for the database. Then, most of the

processing time during online test usage is spent on computing correlations between the convolutional features. The implementation for computing correlations has not been extensively optimized. Due to their limitations, using the correlation and classification layers might not even be the best method for matching. Alternative matching methods might be difficult to use in SGD training that requires computing the gradients of all calculations. Our implementation is simple to use in SGD training, and it allows finetuning the convolutional parameters, which is very important according to our results. However, the matching approach could be replaced with something better or more efficient while using the trained network since gradients are not needed during online identification.

VI. Conclusions

This paper develops a two-stream convolutional neural network for log identification. Logs are matched based on the correlations of deep features computed for the example and database frames. The developed model performs very well for both spruce and pine logs; a single image of the trunk is typically sufficient for identifying a log from a set of thousands of log images. Although it is necessary to improve the methods to generalize the results and to run additional tests with more comprehensive datasets, the results suggest that the model could be applied in challenging real-world forestry conditions, in various applications of log identification.

Acknowledgment

We thank Stora Enso Oyj, specifically Mr. Teppo Salmi, and other parties of the LogID project for useful discussions and for providing the dataset of tree logs.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS) 25*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2014, pp. 580–587.

[3] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015, pp. 3431–3440.

[4] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems (NIPS) 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 568–576.

[5] J. Bromley, J. W. Bentz, L. Bottou, I. Guyon, Y. Lecun, C. Moore, E. Säckinger, and R. Shah, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 07, no. 04, pp. 669–688, 1993.

[6] S. Chopra, R. Hadsell, and Y. LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, Jun 2005, pp. 539–546.

[7] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Springer International Publishing, 2016, pp. 749–765.

[8] R. Tao, E. Gavves, and A. W. M. Smeulders, "Siamese instance search for tracking," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016, pp. 1420–1429.

[9] Q. Guo, W. Feng, C. Zhou, R. Huang, L. Wan, and S. Wang, "Learning dynamic siamese network for visual object tracking," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 1781–1789.

[10] B. Li, J. Yan, W. Wu, Z. Zhu, and X. Hu, "High performance visual tracking with siamese region proposal network," in *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2018.

[11] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. v. d. Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015, pp. 2758–2766.

[12] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *Computer Vision – ECCV 2016 Workshops*, G. Hua and H. Jégou, Eds. Springer International Publishing, 2016, pp. 850–865.

[13] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015, pp. 4353–4361.

[14] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "Matchnet: Unifying feature and metric learning for patch-based matching," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015, pp. 3279–3286.

[15] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016, pp. 5695–5703.

[16] J. Žbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2015, pp. 1592–1599.

[17] Z. Zheng, L. Zheng, and Y. Yang, "A discriminatively learned cnn embedding for person reidentification," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 13:1–13:20, Dec. 2017.

[18] W. Li, R. Zhao, T. Xiao, and X. Wang, "Deepreid: Deep filter pairing neural network for person re-identification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2014, pp. 152–159.

[19] M. Vihlman, H. Hyyti, J. Kalmari, and A. Visala, "Detection and species classification of young trees using machine perception for a semi-autonomous forest machine," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 1543–1548.

[20] S. Bertrand, R. B. Ameur, G. Cerutti, D. Coquin, L. Valet, and L. Tougne, "Bark and leaf fusion systems to improve automatic tree species recognition," *Ecological Informatics*, vol. 46, pp. 57–73, 2018.

[21] S. Boudra, I. Yahiaoui, and A. Behloul, "A comparison of multi-scale local binary pattern variants for bark image retrieval," in *Advanced Concepts for Intelligent Vision Systems*, S. Battiato, J. Blanc-Talon, G. Gallo, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer International Publishing, 2015, pp. 764–775.

[22] T. Ojala, M. Pietikäinen, and D. Harwood, "A comparative study of texture measures with classification based on featured distributions," *Pattern Recognition*, vol. 29, no. 1, pp. 51–59, 1996.

[23] S. H. Lee, C. S. Chan, S. J. Mayo, and P. Remagnino, "How deep learning extracts and learns leaf features for plant classification," *Pattern Recognition*, vol. 71, pp. 1–13, 2017.

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[25] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "Kaze features," in *Computer Vision – ECCV 2012*, A. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds. Springer Berlin Heidelberg, 2012, pp. 214–227.

[26] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *ACM International Conference on Multimedia (MM)*, 2014, pp. 675–678.

[27] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2016, pp. 770–778.

[28] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, Jun 2005, pp. 886–893.