An ensemble of Deep Convolutional Neural Networks for Marking Hair Follicles on Microscopic Images

Łukasz Podlodowski, Szymon Roziewski, Marek Nurzyński
National Information Processing Institute
Natural Language Processing Laboratory
al. Niepodległości 188b 00-608 Warsaw, Poland
Email: lpodlodowski@opi.org.pl, sroziewski@opi.org.pl, mnurzynski@opi.org.pl

***Abstract*—This paper presents an application of a Convolutional Neural Network as a solution for a task associated with ESENSEI Challenge: Marking Hair Follicles on Microscopic Images. As we show in this paper quality of classification results could be improved not only by changing architecture but also by ensemble networks. In this paper, we present two solutions for the task, the first one based on benchmark convolutional neural network, and the second one, an ensemble of VGG-16 networks. Presented models took first and third places in the final competition leaderboard.**

I. Introduction

IN THIS paper, we present our solution to the ESENSEI Challenge: Marking Hair Follicles on Microscopic Images. We used Deep Convolutional Neural Networks (DCNNs) models [1] to solve the given problem. Our final result consisting of an ensemble of DCNNs models was the best performing solution of the challenge. DCNNs training was a massive task, which took a few days using GPU based computation. Before that, we had tested different architectures and did many experiments in order to find a better approach. Subsequently, we trained derived architectures with gentle modification applied. After all, we prepared an ensemble which achieved better score with F-measure. In this paper, we also present 3rd place solution of the challenge based on benchmark CNN model.

II. Related work

We found one study dealing with intelligent image analysis of hair follicles. In [2], advanced automatic target recognition (ATR) algorithms for identification of hair follicles were used. The approach was not related to neural networks, though.

Instead of learning convolutional filters, the authors used the ATR method which performed wavelet filtering of the targets for enhancing the contrast of hair features in the images. Subsequently, the collected features were sent to Adaboost classifier for training and recognition. The system itself was able to provide the intended hair follicle locations. The follicle locations were isolated by continuous wavelet transform in a grayscale image. The authors observed that plotting a grayscale picture of skin with follicles in three dimensions, lead to a print with local minima in the surface plot, in which hair follicles resided. The cross-section of a hair follicle resembled an inverted version of the second derivative of a Gaussian wavelet, referred as Mexican hat wavelet. The correlation between both, cross-section and a Mexican hat wavelet was used to locate a hair follicle. Finally, correlations were used to find regions of interest from the image. Subsequently, features for follicles identification were extracted e.g. entropy, skewness, mean, standard deviation, feature area, minimum value, correlation peak and Euler Number.

The first stage of the ATR system was the optical correlation operation, with the intentionally low set threshold. The output contained many false positives, which were reduced in the second stage, by application of adaptive boosting. The lower bound of prediction performance was 77 % with a maximum of 2 false positives per image. However, the images used in [2] were not containing any hair, but only hair follicles, which seemed to be a fairly easier task to solve.

III. Problem description

The dataset consisted of microscopic digital images of a human scalp. The picture was in full HD resolution, with size 1920x1080 pixels. Each image was divided into equal non-overlapping 144 sectors. Each sector forms a subimage square with size 120x120 pixels. It means that each original image was made of 144 bitmaps, framed in 16 columns and 9 rows. Those subimages represented an defined input of classification problem and contained the visual representation of possible the hair follicle occurrence. In another bitmaps file, the knowledge reflecting the presence of follicles was collected. If a subimage contained at least one follicle, then it was marked by 1, otherwise by 0. Finally, the task was to classify all sectors from an image.

The training set contained 4880 full HD pictures in jpeg format, with corresponding bitmaps containing encoded follicle presence. However, dividing this images into 144 sectors provided 702720 training samples. Approximately 14% of this samples had been labeled as positive.

The test set was made of 1000 images. The task was to predict the follicle positions in the test pictures by classification of each sector. Each prediction for an image had to contain

144 binary values, concerning 16x9 square areas making one image entity. Finally, 144000 test samples were classified.

IV. Data processing

A. *Data preprocessing*

We applied some preprocessing steps in order to improve quality of classification. Those steps were performed on both sets, training and test. We treated all pictures as one channel representing an image in grayscale. We experienced that using 3 color channels was not worth all the effort needed, greatly increasing input data size without significant result improvement. All image data were standardized with respect to the training and test set. Since the images were with size 120x120 pixels, we scaled them to lower resolutions for different models, i.e. 24x24, 36x36 and 48x48 pixels. We were concerned about prediction quality on the sector edges. We also recognized classified that a sector could strongly rely on information from neighbour sectors, i.e. recognizing hair geometry in neighbourhood could provide important information about hair facility occurrence in the analyzed sector. In order to make it more robust, we merged near neighbourhood areas with analyzed subimage, making one larger new input image, to better recognize follicles on the image boundaries. We used a few approaches to this problem, one of them used whole neighbouring sectors. As result, we used area of 9x9 sectors, where classified sector was always in the center of the area. In this way, we created new datasets, with different surrounding cell sizes. The first set was made of images in size 24x24 pixels, joined with their 8 neighbours (also 24x24 px). As a result, we got 72x72 pixels images. We obtained further two datasets, but in this case, we limited the surrounding area by lowering the size of the neighbouring squares by factor 2. Thus, we produced new input images with sizes 72x72 and 96x96 pixels. For the boundary images, when no neighbours were available, we applied images filled with zeros.

B. *Data augmentation*

Data augmentation is a very common technique in image classification, used for the training dataset enlargement. What we had to do, was to perform minor alterations of existing images. Those minor changes might be following: translations; rotations; both, horizontal and vertical flipping; zooming or more sophisticated transformation. The neural network would treat such generated picture as a distinct one, and thus benefit from it. ”Data augmentation has been shown to produce promising ways to increase the accuracy of classification tasks” [3]. This technique ”has played an active role in achieving state-of-the-art results on many vision tasks” [4].

V. Architecture of network

We used two different architectures for each solution. In this section, we describe both of them and final models based on a benchmark CNN and an ensemble of deeper CNNs based on VGG-16 architecture. We also want to discuss the problem of damping position information in CNN.

A. *Position information in CNN*

As we said, our input data included neighbour sectors which did not represent directly visual information about the classified sector. It was very important to keep information about that signal of hair facility occurrence raised from central sector of input data. CNN could damp this information by their architecture nature. Convolutional filters are applied to the whole image and allow to propagate information about pattern occurrence in the specific area. However, the output of filters is normally processed by pooling layer which makes ”mixture” of information from nearest sectors. In the most common approach, the max signal value is chosen from this area. A solution to this problem could be removing pooling layers from a network. However, pooling layers allow to greatly decrease the number of parameters in the network, reducing the size of an analyzed topology of outputs from the previous convolutional layer. This property of pooling is very desired in CNN and allow to greatly simplify model complexity. As shown in [5], in some cases CNN could not include pooling layers and perform parameter reduction by applying a bigger stride of a filter. Whereas, it can perform with similar results as classical CNN architecture with pooling layer. This approach allows keeping pattern occurrence position information through whole signal propagation. However, the bigger stride could also cause skipping some patterns in the analyzed area. Stride-based architecture could also have the problem with generalization made network rigidly associated with input data pattern positions, which in turn, could lead to generalization problem. Based on this information we decided to use average pooling, which does not damp position information from the specific area like max pooling does but make this information as a mixture from neighbour areas which finally provide noised information about pattern occurrence position.

The problem of pattern position also limited possible data augmentation techniques, i.e. shifting could noisy relation between the input and the desired output.

B. *VGG-16*

We made use of cross-validation to choose better models and to estimate learning rate parameter. Finally, we have experienced that a VGG-16 [6] neural network is outperforming other architectures. We applied many experiments with different dense layers, class weights and image sizes. Besides the architecture, the most important factor, in well-performing prediction, is image size. We have observed that F1-score value grows in general, with image resolution. Higher resolutions lead to better results. On the other hand, increasing image size demands more memory in a graphics card and slows down the training process. Likewise, training of a high-resolution model demands more data in general. Thus, we had to balance all that criteria to build a suited and well-performing model. All in all, we built on top of solution an ensemble model made from four models.

The VGG-16 architecture consisting of 23 layers is shown in the Figure 1.

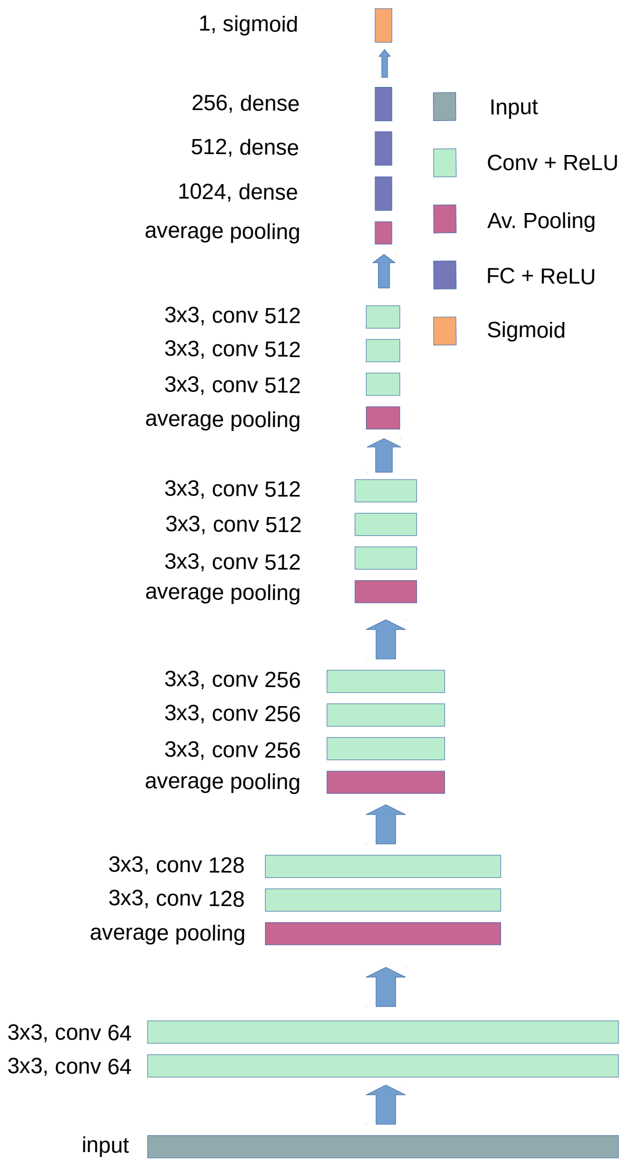

Fig. 1. The architecture of VGG-16 deep convolutional neural network. Different input sizes were used. Before each of convolution and average pooling layers, zero-padding and batch normalization were applied, respectively. After each VGG-16 section, we performed dropout layers to avoid over-fitting. Regarding dense layers, we added dropout and batch normalization after each dense layer.

Instead of max pooling, we used the average, which seemed to perform a bit better. This decision had also analytic background associated with information about pattern position in the image.

We have also modified the number of dense layers, we have four of them. We used dropout to prevent model from overfitting [7] and batch normalization technique [8] to scale and adjust activation functions outputs. Zero padding technique removes unwanted boundary effects. As a loss function, we adapted binary cross-entropy with Adam optimizer for gradient descent optimization.

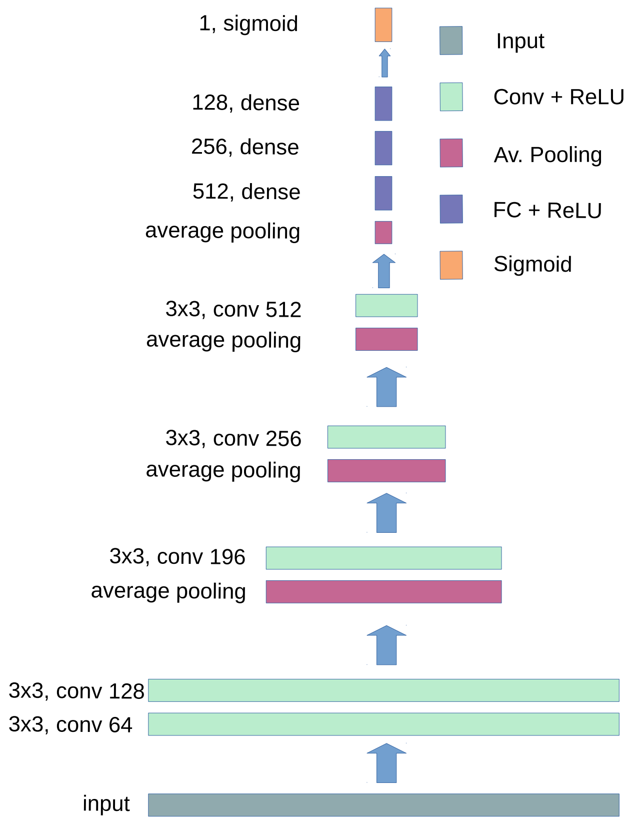

Fig. 2. The architecture of benchmark CNN used as one of the solutions for a defined problem.

VI. Models

Our solution are based on convolutional neural network which are the state-of-the-art models for many image classification problems[9][10][11]. For training networks, we used Adam optimizer[12] which ”has been immensely successful in development of several state-of-the-art solutions for a wide range of problems” [13].

A. *Training*

We applied the learning rate of 0.005 for all models. We have used Adam optimizer for optimization, β_1 and β_2 parameters were set close to 1. This method allows to easily adapt to data and model architecture without a rich parametrization. Networks were trained on 90 percentages of data, rest of set was used for model validation. The early stopping technique was used based on a progress of evaluated on validation data F1-score metric value. Network training optimized binary cross-entropy which do not correspond directly to F1-score. However, we could decrease the gap between both optimization goals by manipulation class weights which could allow us to find the balance between optimizing precision and recall. Different strategies were chosen for different networks. Figure 3 presents metrics values during a training process. It is noticeable that shallow network train is more stable, with fewer outliers points. The mean values of loss functions and f1 scores for models m_1, m_2, m_3, m_4 for both training and validation sets are given in tuples respectively: $(0.341 \pm 0.052, 0.715 \pm 0.035)$ and $(0.214 \pm 0.026, 0.715 \pm 0.023)$. One can observe the equality of F1-scores for training and validation sets.

B. Benchmark CNN

At first, we want to describe CNN architecture which took 3rd place in the competition. For convenience, we will call this model as benchmark CNN. Figure 2 presents the architecture of model. To prevent this network from an overfitting problem, between all convolutional layers we used a dropout with $p = 0.4$, for fully connected layers $p = 0.5$. Batch normalization allowed to damp too strong signals spikes. This network was processing images with 48x48 pixels representing an area of 9x9 sectors. The classified sector was always in the center of the area. Class weights were fixed as 3 per positive class and 1 per class negative class. This value was adjusted experimentally. In next sections, we will discuss and compare results of this model with an ensemble of VGG-16 networks. In our experiments, we treat the results of this model as a benchmark. The model consists of 2,3 mln trainable parameters.

C. DCNN ensemble

As we stated before, we trained 5 models with respect to the given DCNN architecture in Table 1. As shown in [14] ensemble of CNN could enhance results quality. For convenience, we will call members of ensemble m_1, m_2, m_3, m_4, m_5.

Our two models m_1, m_2 have 72x72 pixels images on input. The input image construction has been mentioned in Section IV. Basically, one image is taken from 9 original subimages. Those models differ only in class weights. Although there are almost 6 times more negative samples than positive, we found that the class weights 2 or 3 perform better. The first model has class weights 1 and 2, and the second one has 1 and 3, where the first weight is for class 0 and second for class 1. We monitor the training process to not overfit the model, by using early stopping method. We feed the model under training with a validation dataset, in order to stop training if the model performance measured in F1-score is getting lower, while we keep the best performing model at hand. Both of the models contained roughly 17.5 mln trainable parameters each. The validation set was a 10% subset of total training data.

Subsequently, we have a model m_3 with input images 72x72 pixels, but with higher resolution. The central cell is made from 36x36 pixels image. We kept the same setup and applied class weights 1 and 3. This model has the same number of parameters to estimate. The outcome was slightly better than in the previous approach.

In the end, we trained a model m_4 with the highest resolution. The training itself was a massive task. Since we applied the input images with 96x96 pixels, we had to compute values of more than 20 mln parameters. We trained this model like others with using early stopping. But this time we wanted to re-train a new model m_5 from scratch, with make use of

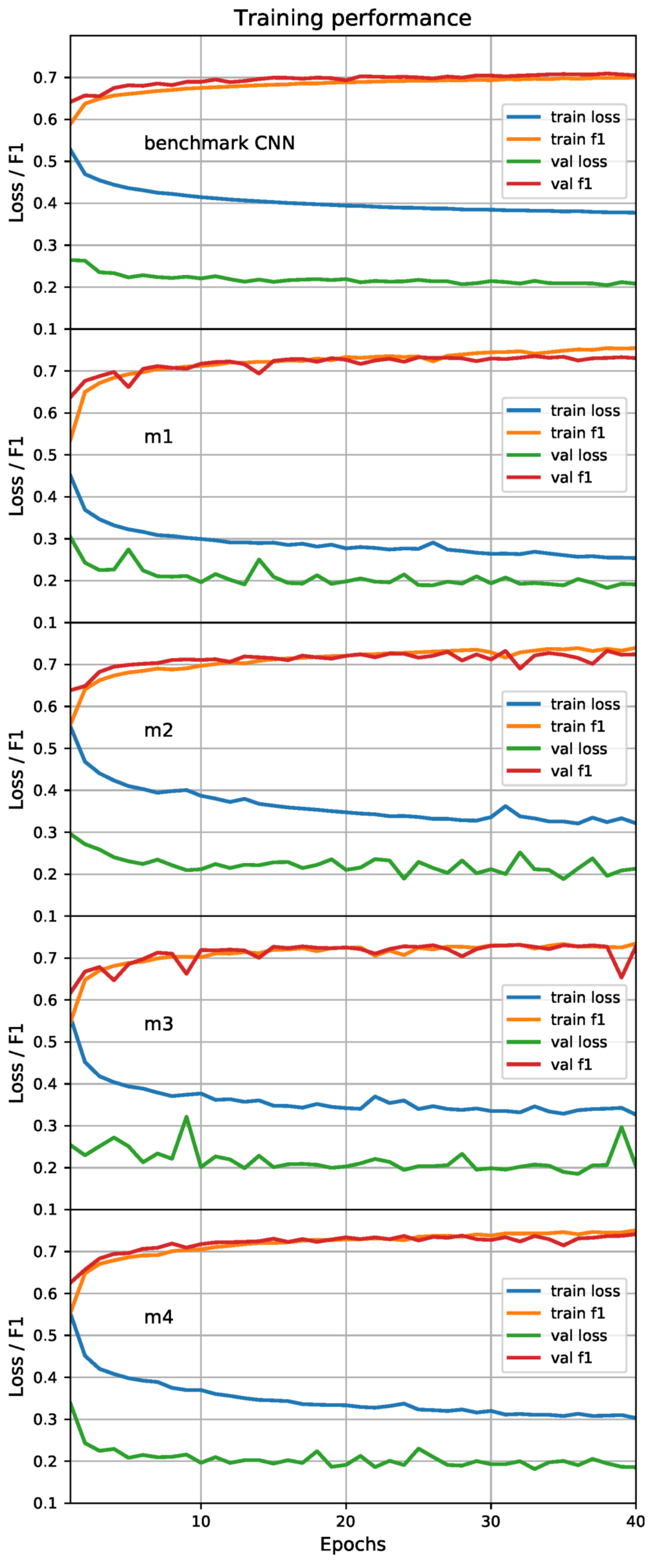

Fig. 3. Training performance charts, for each models we present both, loss function and F1 score performed on training and validation data sets.

all data, including validation data. During the first training, we estimated the number of epochs needed to establish the parameter. Then, in the next round, we just added 2 epochs more and started computing.

In the end, we made an ensemble of those described 5 models, which performed nicely on the leaderboard.

In the Figure 3 shown measurements from a training process for all models from the ensemble. F1 score values and binary cross-entropy loss function values seem to not always been correlated in outliers points. However, trends on both function are similar. We will discuss this problem in the next section.

VII. Results

Our models m_1, m_2 are trained for more than 30 epochs, each epoch lasts for about 33 minutes. F-measure calculated on the validation data set is about 0.736. The model m_3 has a bit higher F1 score – 0.738. The last model m_4 has been trained for 37 epochs with F-measure 0.741. The epoch time length was close to 54 minutes on GPU P40 machine. It means that training of the biggest network took more than 33 hours of excessive computations.

After all, we created an ensemble of 5 models: m_1, m_2, m_3, m_4, m_5 by computing arithmetic average based on predicted probabilities of the follicle presence.

TABLE I
Experimental results - classification quality based on crossvalidation

Model	F1 score	Binary cross-entropy loss
benchmark CNN	0.7132	0.2116
m_1	0.7357	0.1859
m_2	0.7329	0.1960
m_3	0.7384	0.1913
m_4	0.7415	0.1858

As shown in table I and fig. 5 m_1's loss function is almost equal to m_4's and m_3's loss function results. However F1 score difference between this models is noticeable (fig. 4). The main reason for this behaviour is associated with lack of direct relation between F1 score and optimized binary cross-entropy function. F1 score could be seen as second level training quality function. It is not directly implied by cross-entropy but strongly correlated. This correlation is depended on a prior distribution of classes in training set. However, we could influence on strength of this correlation by changing class weights. The training set was unbalanced and contained approximately 14% of positive classes samples. This state leads optimization process to focus on precision of positive class classification. Increasing positive class weight leads model to choose a positive class as an output more frequently which implies increasing a recall importance in an optimization process. F1 score is the harmonic average of the precision and recall, which means that desired optimization method should focus equally on precision and recall to reach the best results.

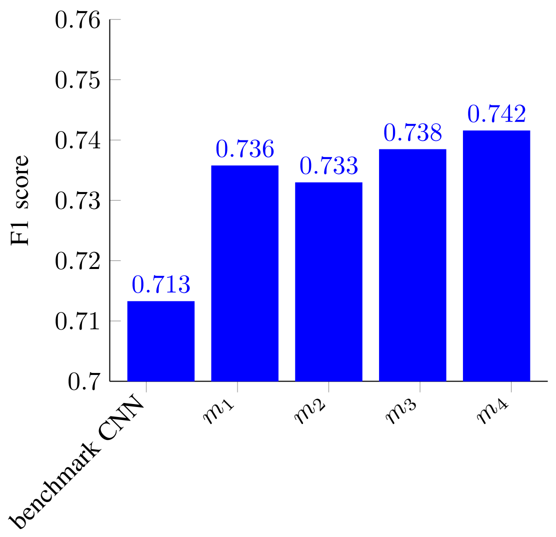

Fig. 4. F1 score results for specific models based on 3-fold cross-validation

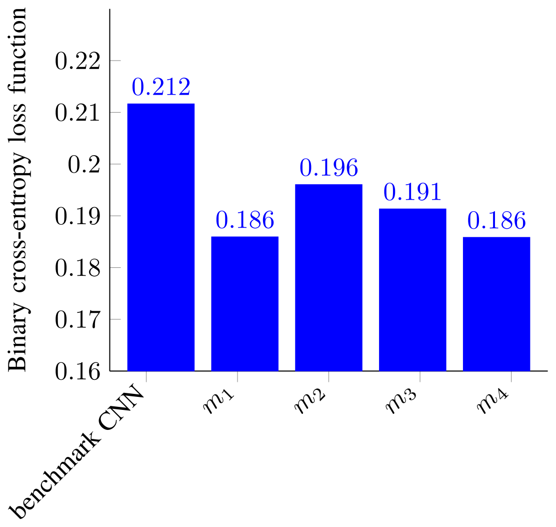

Fig. 5. Binary cross-entropy loss function results for specific models based on 3-fold cross-validation

The second reason could be damping output signal. A network could prefer to minimizing error penalty by taking less risk in making the decision and return probabilities near threshold value of 0.5. This behaviour could have many reasons. First of them could be wrong labeling in training set, which forces the network to generate different output from almost the same inputs. The second one could be a too limited capacity of the model to handle data and force network to bring outputs closer to a prior distribution. A possible solution for increasing a quality of the final class prediction could be changing a threshold or manipulating class weights. However, class weights manipulation had the influence on an entire training process which allows us to use directly optimization process as a tool to solve problems with unbalanced data. This the reason we choose this option to handle this problem.

The results achieved by VGG-16 models outperform benchmark CNN model which shows that deeper network architecture allows to better solve this classification task. Finally, an ensemble of VGG-16 models improve classification quality and took first place in the competition with the final score of 0.763.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85 – 117, 2015. doi: https://doi.org/10.1016/j.neunet.2014.09.003

[2] B. Walker, T. Lu, and T.-H. Chao, "Intelligent image analysis for image-guided hair removal and skin therapy," vol. 8207, p. 820707, 2012. doi: 10.1117/12.910741. [Online]. Available: http://adsabs.harvard.edu/abs/2012SPIE.8207E..07W

[3] L. Perez and J. Wang, "The effectiveness of data augmentation in image classification using deep learning," *CoRR*, vol. abs/1712.04621, 2017. [Online]. Available: http://arxiv.org/abs/1712.04621

[4] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, "Adaptive data augmentation for image classification," 01 2016.

[5] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. A. Riedmiller, "Striving for simplicity: The all convolutional net," *CoRR*, vol. abs/1412.6806, 2014. [Online]. Available: http://arxiv.org/abs/1412.6806

[6] W. D. Shuying Liu, *Very deep convolutional neural network based image classification using small training sample size*. IEEE, 2015.

[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2627435.2670313

[8] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

[10] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings.*, Aug 2003. doi: 10.1109/ICDAR.2003.1227801 pp. 958–963.

[11] D. C. Cireşan, U. Meier, J. Masci, L. M. Gambardella, and J. Schmidhuber, "Flexible, high performance convolutional neural networks for image classification," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence - Volume Volume Two*, ser. IJCAI'11. AAAI Press, 2011. doi: 10.5591/978-1-57735-516-8/IJCAI11-210. ISBN 978-1-57735-514-4 pp. 1237–1242. [Online]. Available: http://dx.doi.org/10.5591/978-1-57735-516-8/IJCAI11-210

[12] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[13] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=ryQu7f-RZ

[14] M. Izadyyazdanabadi, E. Belykh, M. Mooney, N. Martirosyan, J. Eschbacher, P. Nakaji, M. C. Preul, and Y. Yang, "Convolutional neural networks: Ensemble modeling, fine-tuning and unsupervised semantic localization," *CoRR*, vol. abs/1709.03028, 2017. [Online]. Available: http://arxiv.org/abs/1709.03028