Dog Breed Identification Using Deep Learning

Zalán Ráduly
Babeş–Bolyai University
Cluj-Napoca, Romania
raduly.zalan@gmail.com

Csaba Sulyok
Babeş–Bolyai University
Cluj-Napoca, Romania
csaba.sulyok@gmail.com

Zsolt Vadászi
Codespring
Cluj-Napoca, Romania
vadaszi.zsolt@codespring.ro

Attila Zölde
Codespring
Cluj-Napoca, Romania
zolde.attila@codespring.ro

***Abstract*—The current paper presents a fine-grained image recognition problem, one of multi-class classification, namely determining the breed of a dog in a given image. The presented system employs innovative methods in deep learning, including convolutional neural networks. Two different networks are trained and evaluated on the Stanford Dogs dataset.**

The usage/evaluation of convolutional neural networks is presented through a software system. It contains a central server and a mobile client, which includes components and libraries for evaluating on a neural network in both online and offline environments.

***Index Terms*—convolutional neural networks, dog breed identification, fine-grained image recognition, image classification, Inception-ResNet-v2, mobile trained model, Stanford Dogs dataset**

I. Introduction

Nowadays, convolutional neural networks (CNN) [1] are popular in different topics of deep learning: image recognition [1], detection [2], speech recognition [3], data generation [4], etc.

Several traditional image recognition methods are known: Scale-Invariant Feature Transform (SIFT) [5], Histogram of Oriented Gradients (HoG) [6], attribute classification with classifiers: Support Vector Machine (SVM), Softmax and Cross Entropy loss. However, CNNs have also gained significant traction in this field in recent years, mostly due to general reoccurring architectures being viable for solving many different problems.

The current paper presents the methodology and results of fine-tuning CNNs for two different architectures, using the Stanford Dogs dataset [7]. This constitutes a classification problem, but also one of fine-grained image recognition, where there are few and minute differences separating two classes.

Convolutional neural networks are very similar to Artificial Neural Networks [8], which have learnable weights and biases. The difference is the filters, which process over the whole image and are effective in image recognition and classification problems. Deep CNNs are viable on large dataset [9] and are even accurate in large-scale video classification [10]. Fine-tuning methods and learning results for the Inception-ResNet-v2 [11] and NASNet-A mobile [12] architectures are presented. Furthermore, the usage of the trained convolutional neural networks is visualized through a separate software system employing modern technologies. This system is able to determine the breed of a dog in an image provided by the user, and also displays detailed information about each recognized breed. It consists of two main components: a mobile client and a centralized web server.

The remainder of the document is structured as follows: Section II provides an overview of similar approaches in the literature, while Section III presents the used and preprocessed Stanford Dogs dataset. Section IV details the learning of two different CNNs, with Section V encapsulating the results thereof. Providing a practical edge to these CNNs, the accompanying software system is described in Section VI. Conclusions are drawn and further development plans are proposed in Section VII.

II. Related works

The current section presents previous attempts at addressing the problems tackled by the current research. Abdel-Hamid et al. [3] solve a similar problem, one of speech recognition, using traditional methods, making use of the size and position of each local part, and PCA, while Sermanet et al. [13] and Simon et al. [14] mention convolutional neural networks with different architectures.

Liu et al. [15] present alternative learning methods in 2016 using attention localization, while Howard et al. in 2017 [16] present a learning of a CNN using the MobileNet architecture and the Stanford Dogs dataset extended with noisy data.

Similar fine-grained image recognition problems are solved by detection. For example, Zhang et al. [17] generalize R-CNN to detect different parts of an image, while Duan et al. [18] discover localized attributes. Angelova et al. [19] use segmentation and object detection to tackle the same issue. Chen et al. [20] use selective pooling vector for fine-grained image recognition.

The current research is based on fine-tuning CNNs and the results thereof are reproducible on the original Stanford Dogs dataset using the presented methods.

III. Dataset and Preprocessing

The presented CNN learning methodology revolves around the Stanford Dogs [7] dataset. It contains 120 different dog breeds and is a subset of the more general ImageNet [21]. It is separated into training and test dataset. Both sets contain images of different sizes and every image is given a label representing the embodied dog breed. The training dataset contains 12.000 images with roughly 100 per breed; the test data consists of 8.580 unevenly distributed images.

978-1-5386-6841-2/18/$31.00 ©2018 IEEE

The first step of the preprocessing is to split the training data into train folds and validation fold for experimental tuning of the learning hyperparameters. Before splitting the data, the dog images are resized to 256x256 pixels (for NASNet-A mobile) and 299x299 pixels (Inception-ResNet-v2 input).

For experimenting with the hyperparameters of the CNNs, fine-tuning and 5-fold cross-validation is used, which in the end produces 5 different training and validation subsets. Each of these datasets contains 9.600 training images and 2.400 validation data. After getting the best fine-tuned hyperparameters, the entire Stanford Dogs training dataset is used for training, while the test data is exclusively used for evaluation.

IV. Experiments

After obtaining the necessary formatted and resized data, the next step is fine-tuning the convolutional neural networks. This section presents the applied technologies, CNN architectures, methods, hyperparameters and using the trained models as frozen graph.

The presented problem fits into the category of fine-grained image recognition, since the differences linking any sample image to a certain class are few and minuscule; the CNN must consider small key features to dissolve ambiguity. For example, the husky and the malamute breeds present small enough differences among them to make differentiating difficult even for trained eyes.

A. CNN Architectures

Transfer learning [22] provides a performance boost by not requiring a full training from scratch. Instead, it uses pre-trained models which are taught general reoccurring features. These models are often trained on the ImageNet [21] dataset, which has a competition every year and some pre-trained models are published. The learning of these models represents fine-tuning the given dataset with the learned weights and biases. The current research contains two different public pre-trained convolutional neural networks, which are fine-tuned: NASNet-A mobile [12] and Inception-ResNet-v2 [11].

The NASNet-A architecture is created based on the approach of the Neural Architecture Search (NAS) framework [23], by the Google AutoML [24].

The Inception-ResNet-v2 is a very deep architecture containing over 300 layers, which is created by the Google developers team.

B. Data augmentation

The most common method to reduce overfitting on training data is to use different transformations before the feedforward pass during the training; this is called data augmentation [9].

During the learning of the CNN models, another preprocessing method, augmentation, is applied. The Inception-ResNet-v2 and the NASNet-A mobile architecture uses Inception preprocessing, which is the following function:

$$f(x) = (\frac{x}{255.0} - 0.5) * 2.0$$

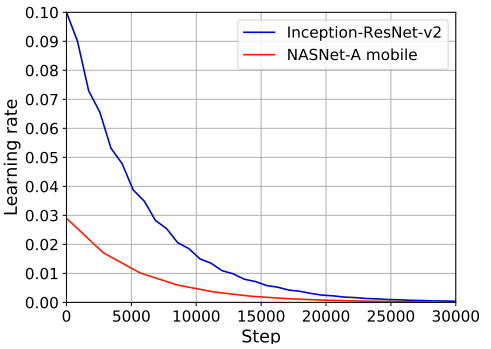


Fig. 1. The adaptive learning rate during the training sessions decay exponentially by 10% every 3 epochs. The blue is the learning rate of the Inception-ResNet-v2 with an initial value of 0.1, while the red is the NASNet-A learning rate with an initial value of 0.029.

where x the image. Before the Inception preprocessing, the image is randomly reflected and cropped by the TensorFlow's distorted bound box algorithm.

For the evaluation of the validation or the test dataset, the applied augmentation steps include an 87.5% central crop and the Inception preprocessing.

C. Learning and hyperparameters

The fine-tuning experiments of the convolutional neural networks are performed on a personal computer with a GeForce GTX 1080 GPU, an Intel Core i5-6400 CPU and 64 GB RAM.

During the learning, a Softmax Cross-Entropy loss function and Nesterov momentum [25] optimizers are used for the fine-tuning of the NASNet-A mobile model and the Inception-ResNet-v2 model. During the training, the last fully-connected layer (logits) is unfrozen and fine-tuned, while the other layers are unchanged and frozen. This makes use of the benefits of the pre-trained models.

The first parts of the CNN training involve hyperparameter tuning using cross-validation. During the experiment the hyperparameters are chosen empirically. After resulting in the appropriate parameters, another phase of learning begins on the entire Stanford Dogs training dataset, with the model evaluated on the test dataset in this case.

Both convolutional neural network are trained with the following common hyperparameters: a batch size of 64, an exponentially decreasing learning rate with different initial value, where the rate decays 10% every 3 epochs (approximately 563 steps), a 0.0001 weight decay, and training for 30.000 steps. The NASNet-A mobile architecture is fine-tuned with a learning rate with an initial value of 0.029 (see Figure 1). The Inception-ResNet-v2 is fine-tuned with a learning rate with an initial value of 0.1 (see Figure 1). Training the mobile model takes three times less than the Inception-ResNet-v2 model.

Further experimented hyperparameters include: fixed learning rates (0.01, 0.001), exponentially adaptive learning rates (with initial values of 0.031, 0.035), default weight decay (0.0004), different numbers of steps for training (15.000, 20.000, 20.500) and different optimizer (RMSprop).

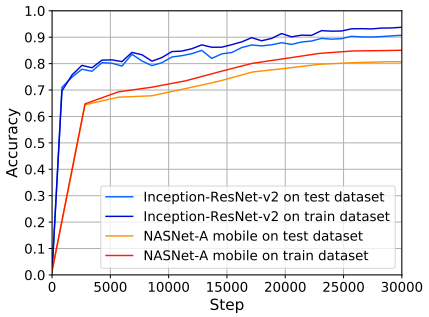
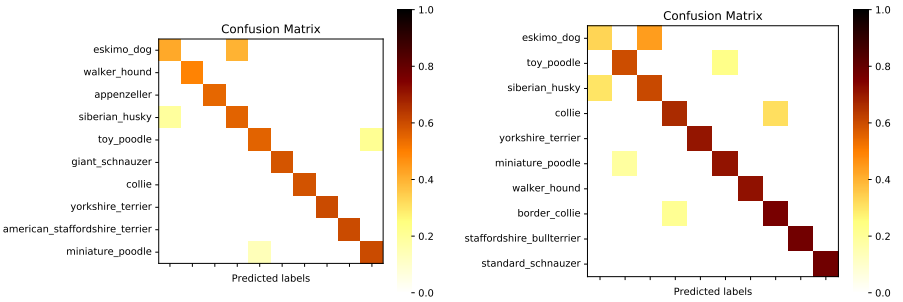


Fig. 2. The accuracy of the trained NASNet-A and the Inception-ResNet-v2 models on the train and test dataset.



(a) NASNet-A mobile bottom 10 classes (b) Inception-ResNet-v2 bottom 10 classes

Fig. 3. Normalized confusion matrices depicting the bottom 10 ranking classes from the test dataset. The y axis holds the actual labels, while the x axis shows the predicted labels in equivalent order.

D. Frozen graph

To use the taught CNN for evaluation through an API, it is necessary in preamble to freeze the model. This step involves freezing, saving and exporting the model of the neural network to a single binary file including the structure and parameters.

V. Results

The trained models are evaluated every 10 minutes on the training and test dataset. In this section, the relevant metrics for evaluation are presented: accuracy (see Figure IV-C), precision, recall (see Table I) and confusion matrices (see Figure 3). The evaluation points are linearly distributed in time, not in number of steps, hence the uneven step distribution.

The accuracy is monitored on the training and test datasets; this metric represents the mean percentage of correctly classified classes on a dataset. The NASNet-A mobile architecture achieves 85.06% accuracy on the train dataset and 80.72% on the test dataset. The accuracy with a deeper CNN shows better results: the Inception-ResNet-v2 network achieves 93.66% accuracy on the train dataset and 90.69% on the test dataset (see Figure IV-C). The superior performance of the latter is not surprising, since it is a much deeper network.

Precision and recall are also measured during the evaluation of the training and test dataset; results are presented in Table I. The precision and the recall for the trained Inception-ResNet-v2 model is better, the deeper model extracts more features from an image and classifies better than the trained NASNet-A mobile model.

Furthermore, confusion matrices are built in the last step of training for both CNNs using the test data, representing the 10 best and 10 worst scoring classes (the bottom 10 are visible in Figure 3). Examining the results, both CNNs have difficulties differentiating between certain classes, e.g. the Eskimo Dog and Siberian Husky, or the Toy Poodle and Miniature Poodle. Even from the confusion matrices, the accuracies may be observed as less for the NASNet-A mobile than for the Inception-ResNet-v2.

The top ranking classes from the confusion matrices are also analyzed. The Inception-ResNet-v2 trained model classifies 10 different dog breeds[1] with 100% accuracy. On the other hand, the NASNet-A trained model classifies only 1 breed with 100% accuracy, namely the Sealyham Terrier.

To gain further insight from the confusion matrices, a statistical analysis is performed, gathering how many different dog breeds are classified correctly between percentage intervals; the resulting histogram is visible in Figure 4. The histogram shows that in the case of the Inception-ResNet-v2 trained

[1]Brittany Spaniel, Chow, Afghan Hound, African hunting dog, Keeshond, Schipperke, Bedlington Terrier, Sealyham Terrier and Curly

TABLE I
Precision and Recall

Metric	*Model*	*Training dataset*	*Test dataset*
Precision	NASNet-A	85.27%	80.03%
Precision	Inception-ResNet-v2	93.86%	90.18%
Recall	NASNet-A	85.06%	79.86%
Recall	Inception-ResNet-v2	93.65%	89.99%

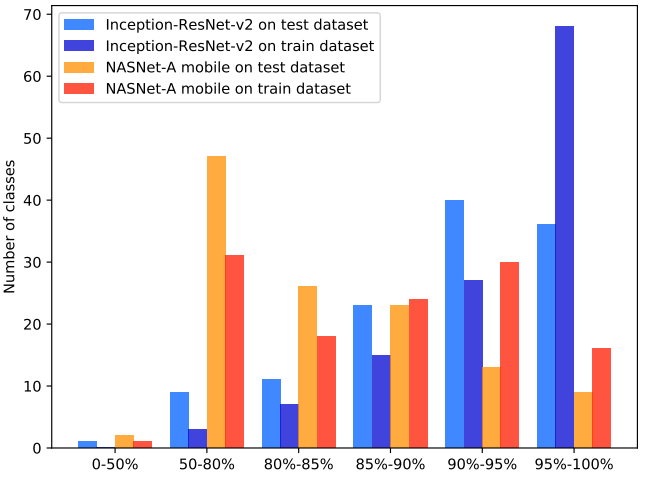


Fig. 4. Histogram of classified class percentages for both CNN architectures on both datasets. A bar represents the count of classes within a range of accuracies.

TABLE II
NASNET-A MOBILE ACCURACY WITH DIFFERENT OPTIMIZERS

Optimizer	*Training accuracy*	*Test accuracy*
Nesterov momentum	85.06%	80.72%
RMSprop	84.94%	80.57%

TABLE III
PERFORMANCE OF SPECIES CATEGORIZATION USING STANFORD DOGS

Method	*Accuracy*
Chen et al. [20]	52.00%
Simon & Rodner [14]	68.61%
Sermanet et al. [13] (GoogLeNet)	75.50%
Krause et al. [26]	80.80%
Liu et al. [15] (ResNet-50)	88.90%
Fine-tuned NASNet-A mobile	80.72%
Fine-tuned Inception-ResNet-v2	90.69%

model, most classes from the train dataset are classified in the (95%, 100%] interval, while from the test dataset the most classes are classified in the (90%, 95%] range. The test dataset is classified well with some inaccuracies remaining. The classified classes from train and test dataset for the NASNet-A model have a wider spread, with a majority of the classes falling in the (50%, 80%] range. The large accuracy difference between the two models is understandably in correlation with the size and complexity of the architectures.

The mentioned accuracies using the NasNet-A architecture are achieved using the Nesterov momentum optimizer. For a comparison, an alternative optimizer, RMSprop, is also tested; the results are similar (see Table II).

Comparison to related work on Stanford Dogs dataset is given in the Table III.

After the evaluation of each trained convolutional neural network, a Grad-CAM [27] heat map visualization is made for the NASNet-A mobile trained model (see Figure 5). The last convolutional block ”pays attention” mostly to the heads of each dogs on an image.

Examining the heat map images, the NASNet-A model mostly focuses on the head of the dogs. In the second image, which shows a German Shepherd in a different position, the CNN pays attention also to the body. If an image contains more than one dog, the network is interested in all of the recognized dogs in varied percentages, and evaluates the image taking each appeared breed into consideration. For an accurate classification it must consider the evaluate an image, which contains different parts of the dog including the head. The displayed images are evaluated correctly by the NASNet-A mobile trained model.

Using different data augmentation: random image rotation between 0 and 15 degrees, random zooming or 87.5% central cropping does not help to improve the accuracy of the trained models.

The alternative hyperparameters presented in Section IV-C (fixed vs. adaptive learning rates, weight decay and step counts) prove to not improve the accuracy or other valuable metrics after training the CNNs.

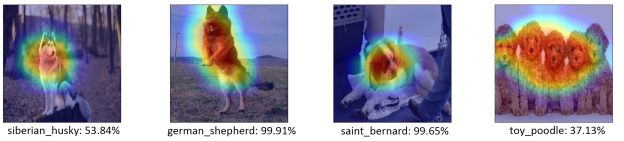


Fig. 5. Grad-CAM visualization with heat maps for the last layers of the NASNet-A mobile trained model. The last convolutional block ”pays attention” to the warm colored parts of the images (mostly to the heads of each dogs), while the cold colors represents the less interested parts of the image. These pictures are not part of Stanford Dogs dataset.

VI. THE SOFTWARE SYSTEM

The usage of the trained convolutional neural networks is presented through a software system, called *Sniff!*. Its associated mobile application gives the opportunity for users to take a photo or select an existing one from the gallery in a mobile application, which not only classifies the image, but also displays detailed information about each evaluated breed. The displayed data serves educational and informative purposes.

The software system consists of two component: a central server written in the Go programming language, and a mobile client realized in React Native. The components communicate via HTTP requests/responses.

The server contains a classifying module using the TensorFlow Go library; it loads the trained convolutional neural network, preprocessing and evaluating images. The Inception-ResNet-v2 and NASNet-A model are both runnable on the server, since a desktop machine can make use of more CPU/GPU resources for a faster evaluation. By default, the Resnet model is used, since it reaches higher accuracies.

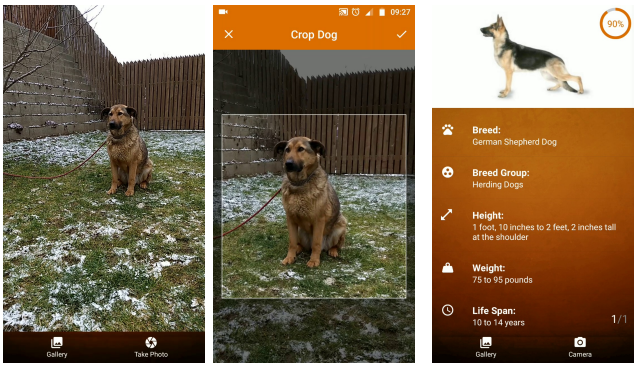
The results of the evaluation for an image are for every classes, the most classes are evaluated with 0% percentage. To avoid the wrong classification with low percentage there is set a `threshold` on the server and also on the mobile client.

The mobile client can take an image with the camera of the smartphone or import one from its gallery, and submit it for classification. The process can happen online using the central server for a faster classification, or offline using the phone resources in case of a lacking network connection. Offline evaluation is facilitated by React Native TensorFlow [28] wrapper library. Figure VI shows the usage of the *Sniff!* application.

The app uses the NASNet-A mobile trained model, which loads every evaluation into the memory of the device; this depends on the resources of the phone. The memory is freed after the evaluation.

The app displays detailed information about the detected breeds, with data web scraped from A-Z Animals[2] and dogtime.com[3].

[2]Source: https://a-z-animals.com/
[3]Source: http://dogtime.com/



(a) Camera (b) Crop (c) Result

Fig. 6. Main components in the Sniff! application.

VII. Conclusions and Future Work

Two different convolutional neural network architectures have been presented: the NASNet-A mobile architecture and the Inception-ResNet-v2 deep architecture.

The architectures have been tested on a niche image classification problem: that of recognizing dog breeds. The pre-trained networks are fine-tuned using the Stanford Dogs dataset.

Results are promising even for the smaller, mobile-friendly CNN, reaching only 10% less accuracy than the deep Inception-ResNet-v2 model.

The usage of the fine-tuned convolutional neural networks is presented through a software system, called *Sniff!*: a mobile application, which can determine the breed of a dog from an image (even without an Internet connection).

The convolutional neural networks can be further developed by: Generative Adversarial Nets (GAN) [4] to extend the training dataset, using other loss function like center loss [29], training other convolutional neural network architectures, expanding the dataset with other popular dog breeds, using detectors for locating multiple dogs on an image and optimizing the server and mobile classification.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems 25*. Curran Associates, Inc., 2012, pp. 1097–1105.

[2] S. Ren, K. He, R. B. Girshick, and J. Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[3] O. Abdel-Hamid, A. r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 22, no. 10, pp. 1533–1545, Oct 2014.

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[5] T. Lindeberg, "Scale Invariant Feature Transform," *Scholarpedia*, vol. 7, no. 5, p. 10491, 2012, revision #153939.

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

[7] A. Khosla, N. Jayadevaprakash, B. Yao, and L. Fei-Fei, "Novel dataset for fine-grained image categorization," in *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO, June 2011.

[8] R. J. Schalkoff, *Artificial neural networks*. McGraw-Hill New York, 1997, vol. 1.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks."

[10] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[11] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016.

[12] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," *CoRR*, vol. abs/1707.07012, 2017.

[13] P. Sermanet, A. Frome, and E. Real, "Attention for fine-grained categorization," *CoRR*, vol. abs/1412.7054, 2014.

[14] M. Simon and E. Rodner, "Neural activation constellations: Unsupervised part model discovery with convolutional networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1143–1151.

[15] X. Liu, T. Xia, J. Wang, and Y. Lin, "Fully convolutional attention localization networks: Efficient attention localization for fine-grained recognition," *CoRR*, vol. abs/1603.06765, 2016.

[16] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *CoRR*, vol. abs/1704.04861, 2017.

[17] N. Zhang, J. Donahue, R. Girshick, and T. Darrell, "Part-based r-cnns for fine-grained category detection," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 834–849.

[18] K. Duan, D. Parikh, D. Crandall, and K. Grauman, "Discovering localized attributes for fine-grained recognition," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, June 2012, pp. 3474–3481.

[19] A. Angelova and S. Zhu, "Efficient object detection and segmentation for fine-grained recognition," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 811–818.

[20] G. Chen, J. Yang, H. Jin, E. Shechtman, J. Brandt, and T. X. Han, "Selective pooling vector for fine-grained recognition," in *2015 IEEE Winter Conference on Applications of Computer Vision*, Jan 2015, pp. 860–867.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[22] Transfer learning and the art of using pre-trained models in deep learning. [Online]. Available: https://www.analyticsvidhya.com/blog/2017/06/transfer-learning-the-art-of-fine-tuning-a-pre-trained-model/

[23] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016.

[24] Google automl. [Online]. Available: https://research.googleblog.com/2017/05/using-machine-learning-to-explore.html

[25] S. Ruder, "An overview of gradient descent optimization algorithms," *CoRR*, vol. abs/1609.04747, 2016.

[26] J. Krause, B. Sapp, A. Howard, H. Zhou, A. Toshev, T. Duerig, J. Philbin, and L. Fei-Fei, "The unreasonable effectiveness of noisy data for fine-grained recognition," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds. Cham: Springer International Publishing, 2016, pp. 301–320.

[27] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," *CoRR*, vol. abs/1610.02391, 2016.

[28] React native tensorflow. [Online]. Available: https://github.com/reneweb/react-native-tensorflow

[29] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *European Conference on Computer Vision*. Springer, 2016, pp. 499–515.