Recognizing military vehicles in social media images using deep learning

Tuomo Hiippala
Digital Geography Lab
Department of Geosciences and Geography
University of Helsinki, Finland
E-mail: tuomo.hiippala@iki.fi

***Abstract*—This paper presents a system that uses machine learning to recognize military vehicles in social media images. To do so, the system draws on recent advances in applying deep neural networks to computer vision tasks, while also making extensive use of openly available libraries, models and data. Training a vehicle recognition system over three classes, the paper reports on two experiments that use different architectures and strategies to overcome the challenges of working with limited training data: data augmentation and transfer learning. The results show that transfer learning outperforms data augmentation, achieving an average accuracy of 95.18% using 10-fold cross-validation, while also generalizing well on a separate testing set consisting of social media content.**

I. Introduction

Just like common moments in everyday life, extraordinary events such as conflicts, crises or periods of increased military tension are likely to be reflected in user-generated content on social media. This content is generated and shared by various parties ranging from media outlets to groups and individuals. Without sufficient operational security in place, user-generated content holds considerable potential for gathering open-source intelligence for governmental, non-governmental and intergovernmental organizations [1].

A prime example of a non-governmental organization is *Bellingcat*[1], which conducts investigative journalism using open sources and social media, publishing meticulous reports on the ongoing conflicts in eastern Ukraine and Syria [2]. Intergovernmental organizations, in turn, may be exemplified by the Organization for Security and Co-operation in Europe (OSCE), which is currently tasked with monitoring the situation in the Donbass region in eastern Ukraine.

Apart from the general goals of organizations mining social media data for open-source intelligence, a crucial difference emerges in their resources for doing such work. Monitoring and analyzing social media content demands extensive manual labour, which may not be available for non- and intergovernmental organizations with limited resources. This is precisely the issue to which this paper contributes by presenting a system for recognizing military vehicles in social media photographs using computer vision and machine learning.[2]

The paper builds on recent advances in the subfield of *deep learning*, which has brought about significant advances in various computer vision tasks, such as object detection, classification and localisation [3]. Keeping the limited resources of non- and intergovernmental organizations in mind, the work leverages the Python machine learning ecosystem for openly available libraries, architectures and pre-trained models.

The paper shows that openly available resources can be used to design an object recognition system that achieves a high accuracy with relatively little training data. Such systems can be used to examine collections of photographs scraped from online sources or plugged into a social media monitoring tools for real-time surveillance. The paper itself is structured as follows: after a brief review of related work, the discussion moves to describe data sources and system design, before evaluating its performance in two experiments. The paper concludes with a discussion and outlines avenues of future research.

II. Related Work

Omand *et al.* [4] propose the term SOCMINT to describe the collection, verification and analysis of intelligence gathered from social media. Identifying the volume of data as a major obstacle to drawing meaningful conclusions from SOCMINT, Omand *et al.* suggest that machine learning could be used to automate aspects of SOCMINT work to assist human analysts. They point towards several areas of research, which they consider applicable to automating the processing of SOCMINT data, such as sentiment and network analysis.

However, one area of research that Omand *et al.* do not mention is computer vision, which has progressed rapidly since 2012, largely due to a renewed interest in neural networks, which play a central role in deep learning. A seminal paper by Krizhevsky *et al.* [5] showed that deep neural networks, which learn the necessary features automatically, outperform manually-created, human-engineered features, which were previously commonly used for describing the content of images. For SOCMINT, deep learning opens up new opportunities for extracting information from visual big data [6], [7].

Deep convolutional neural networks (CNNs) have been used for a wide range of computer vision tasks, including *object recognition*, which aptly characterizes the current task of recognizing military vehicles [8]. To draw on an example from a similar domain, CNNs have been found effective for automatic recognition of different vehicle types. Dong

[1] https://www.bellingcat.com

[2] Available at https://github.com/DigitalGeographyLab/MilVehicles/

et al. [9] show that CNNs can automatically learn features necessary for distinguishing between different vehicle types under different lighting and weather conditions. Their network, which achieves a 92.89% accuracy for a five-way classification task, features a skip architecture that provides both low- (e.g. textures, edges, etc.) and high-level features (e.g. vehicle shapes) learned by the model to the final classifier.

Simpler architectures for CNNs have been shown to perform equally well. In a recent study, Huttunen *et al.* [10] report an accuracy of over 97% for distinguishing between four classes (bus, truck, van and small car) by training a network with two convolutional layers with 32 feature maps each, followed by two dense layers with 100 nodes each, over a data set of 6555 images. The images were collected from two static cameras over a long period of time under differing weather conditions.

In contrast to cars, vans, trucks and buses, however, military vehicles are a relatively rare sight. Consequently, openly available training data is rather scarce, which poses a major challenge, because deep neural networks are notoriously greedy for data, typically requiring thousands of images per class to learn the necessary features. At the same time, military vehicles typically exhibit low intra-class variation, that is, different vehicle types tend to look alike, in contrast to civilian vehicles that come in various forms and sizes. Additionally, neural networks do not have to be trained from scratch, as features learned from other data sets can be transferred across domains using a technique known as transfer learning [11]. This reduces the need for data, whose volume may also be increased using data augmentation. Both transfer learning and data augmentation are explored in the following sections.

III. Data

A. Data Collection

To develop and test the system, training data were collected for three classes with the eastern Ukraine conflict in mind: (1) T-72 main battle tanks, (2) BMP armoured personnel carriers and (3) other images featuring various civilian vehicles, street scenes and everyday images from the conflict zone. Whereas the first two classes, T-72 and BMP, represent tracked vehicles widely used by both sides of the conflict, the third class is essentially a negative class included in the training data to enable the networks to learn about other kinds of photographic material posted from the conflict zone. The images were collected from three sources: (1) photos posted on the social photo-sharing service Flickr, (2) frames extracted from YouTube videos taken in the conflict zone and (3) photos collected from the web. The data sources and volumes are given in Table I.

TABLE I
Data sources and number of images collected

	Flickr	YouTube	Web	Total
T-72	773	513	171	1457
BMP	182	844	62	1088
Other	819	550	108	1477
Total	1774	1907	341	**4022**

As Table I shows, using different sources enabled to roughly balance the amount of data for each class. For instance, although images of BMPs were relatively rare on Flickr, this shortcoming could be remedied by collecting more data from YouTube videos. The videos were sampled at a rate of one frame per second.

Collecting data from the aforementioned sources made manual preprocessing necessary. The Flickr data, for instance, had to be filtered for photographic content, as the users also upload hand-drawn illustrations and photographs of scale models to the service. The YouTube videos posted in the conflict zone were also manually sampled for frames featuring T-72s, BMPs or images of the surrounding landscape. Extracting frames from YouTube videos nearly doubled the volume of data, while also providing images of the vehicles under diverse lighting and weather conditions, and from various angles and distances.

It must be noted, however, that retrieving multiple frames from the same video may populate the data set with images that share many other features besides the vehicles shown in them. Training a neural network over this data runs the risk of overfitting, that is, learning features irrelevant to the recognition task, which needs to be accounted for during training [12].

B. Data Augmentation

Despite the increase provided by sampling video data, the number of images collected was not likely to be sufficient for training a deep neural network from scratch without overfitting the training data. For this reason, more data needed to be synthesized from the available images. This process, which is commonly referred to as data augmentation, involves creating additional copies of the training data by applying random transformations to the images. As Goodwin *et al.* [13, 445] note, data augmentation is particularly suitable for object recognition, as the classes are largely invariant to transformations and may therefore be easily transformed using simple geometric operations.

The *Keras* deep learning library [14] provides an effective solution for online data augmentation, generating batches of augmented data from the source images in real-time during training. The augmentation strategy adopted for this study involved applying random transformations, such as rotations, flips, zooms, shears and shifts, to the source images using the following parameters:

- The images may be flipped horizontally, resulting in a mirrored copy of the original image
- Random rotations up to 10 degrees to either direction, as vehicles are unlikely to appear in more extreme angles
- Random shifts up to 5% of the image height on the vertical axis; 20% of width on the horizontal axis
- Random zooms into the image, up to 120% of the original size, essentially cropping the original image
- Random shears across the image at a maximum angle of 20 degrees

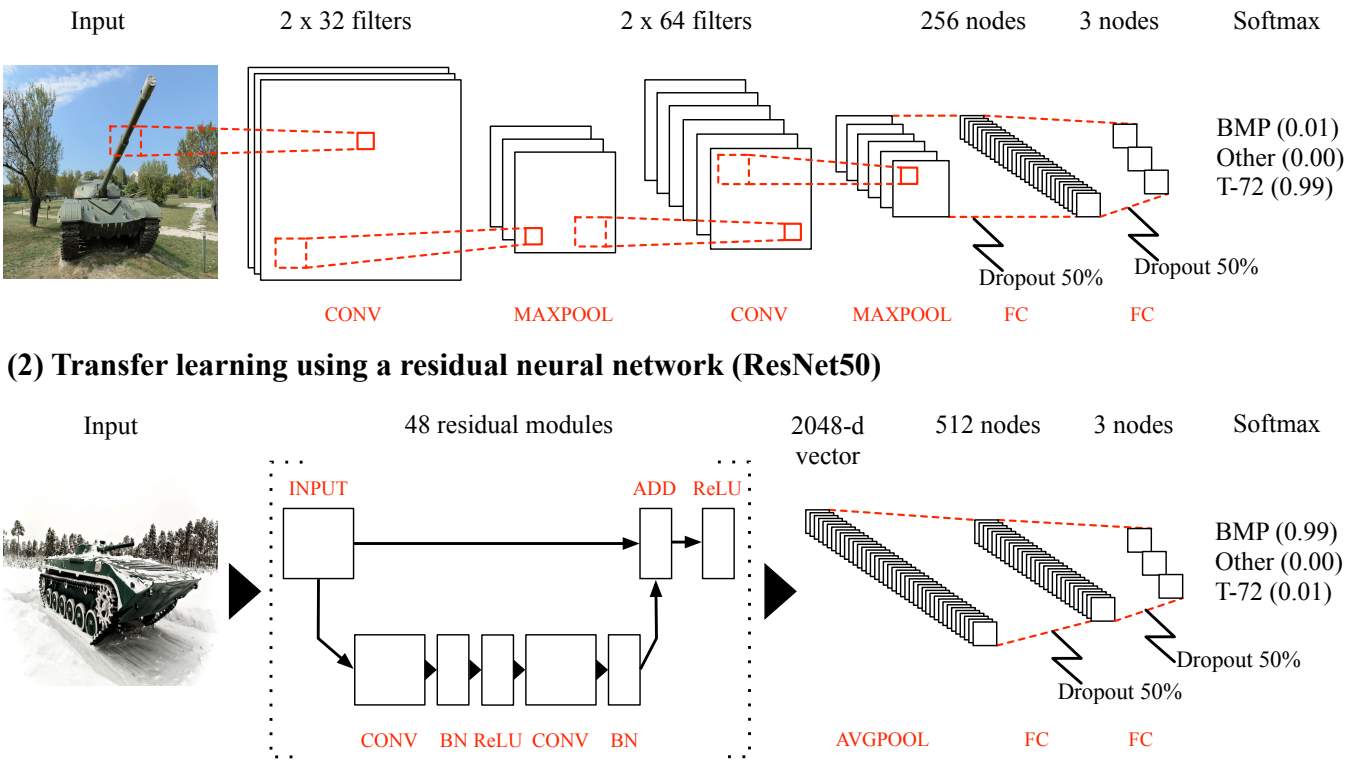

Fig. 1. Network architectures

IV. Network Architectures and Optimization

A. Network Architectures

Two different options were explored for selecting and training a neural network for the recognition task: (1) training a network from scratch using data augmentation to increase the volume of data and (2) training a network without data augmentation, but opting to use transfer learning instead. In this case, transfer learning involved using a neural network pre-trained on data from a different domain to extract features from the data set described in Section III-A, and feeding these features a small, fully-connected neural network [15].

The first option, training a neural network from scratch, involved using a convolutional neural network similar to the one designed for vehicle recognition by Huttunen *et al.* [10], except with additional convolutional layers. This architecture, hereafter referred to as ConvNet, features two convolutional blocks, each consisting of two layers with 32 and 64 feature maps, respectively, followed by a maxpooling operation.

The feature maps are then flattened before a fully-connected layer with 256 nodes, which is followed by a Softmax activation that outputs a probability distribution over the three classes, as shown in Figure 1. To combat overfitting, Dropout [16] is used to randomly disconnect neurons in the fully-connected layer to prevent the network from seeing the same input twice. In addition, the weights for both convolutional and fully-connected layers are penalized using L2 regularisation.

The second option, transfer learning, used a deep residual network (ResNet), a novel architecture which won the 2015 ImageNet Large Scale Visual Recognition Competition (ILSVRC) for classification, detection and localisation tasks [17]. The crucial difference between the convolutional and residual neural networks is that residual networks use a skip connection to add the original layer input to the layer output. He *et al.* show that this kind of architecture facilitates the optimisation of deep neural networks, enabling a considerable increase in network depth. The work on deep residual networks continues, and has yielded even deeper architectures that have been shown to improve performance [18].

Figure 1 illustrates the core element of a residual network, the residual block, which takes the activation from the previous layer as an input. This input is then fed to a convolutional layer, followed by batch normalisation (BN) [19] and a rectified linear unit (ReLU) activation [20]. The number of feature maps learned in each convolutional layer increases with the network depth. Finally, the original activation is then added to the output ahead of another ReLU activation, whose output acts as the input for the next residual block.

ResNets stack multiple residual blocks before performing average pooling, which outputs a 2048-dimensional feature vector that is then fed to a fully-connected layer. Keras [14]

provides 50-layer residual net (ResNet50) out-of-the-box with weights pre-trained on ImageNet [21]. As ImageNet features many different types of vehicles, including main battle tanks and armoured personnel carriers, it may be assumed that the network has learned many of the features relevant for the current recognition task as well. This allows using ResNet50 as a feature extractor, taking the output from the average pooling layer and feeding these features to a fully-connected block to obtain a Softmax probability distribution over the three classes.

B. Hyperparameter Optimization

As said, the experiments consisted of two different scenarios: (1) training a small ConvNet from scratch by taking advantage of online data augmentation, and (2) using the pre-trained ResNet50 to extract features from the original data, before fine-tuning a fully-connected block over the three classes on top.

A random search was used to find optimal hyperparameters for the networks in both experiments, following the approach proposed by Bergstra *et al.* [22] as implemented in the *scikit-learn* library [23]. The random search was run on Keras using Theano backend [24] to overcome memory management problems with TensorFlow. Data augmentation was not used during random search.

For the more computationally expensive experiment (1) with 19M trainable model parameters, 50 random hyperparameter settings were evaluated, training for 50 epochs with 3-fold cross-validation for each setting. For the less expensive experiment (2) with 1M trainable model parameters, 50 random hyperparameter settings were evaluated, training for 100 epochs with 10-fold cross-validation for each setting. Table II shows the randomly searched hyperparameter ranges, which both featured a total of 576 possible settings, while the rightmost column shows the best values found during the random search.

TABLE II
RANDOMLY SEARCHED HYPERPARAMETER RANGES

Architecture	Hyperparameter	Range	Best
(1) ConvNet	Batch size	{32, 64, 128, 256}	256
	Learning rate	{0.1, 0.01, 0.001, 0.0001}	0.0001
	L2 λ	{0.01, 0.001, 0.0001}	0.001
	Dropout rate	{0.25, 0.5, 0.75}	0.5
	Nodes in FC layer	{64, 128, 256, 512}	256
(2) ResNet	Batch size	{32, 64, 128, 256}	32
	Learning rate	{0.1, 0.01, 0.001, 0.0001}	0.01
	L2 λ	{0.01, 0.001, 0.0001}	0.001
	Dropout rate	{0.25, 0.5, 0.75}	0.5
	Nodes in FC layer	{64, 128, 256, 512}	512

V. Experiments and Evaluation

This section describes the training and evaluation of the two network architectures presented above. Both networks were implemented using Keras [14] running on TensorFlow backend [25]. The weight parameters for both networks were optimized using stochastic gradient descent (SGD) with mini-batch sizes given in Table II, a Nesterov momentum of 0.9 and a decay of 10^{-5} to minimize the loss for categorical cross-entropy between the true and predicted labels. The performance of both networks was evaluated using categorical accuracy.

Both experiments used 10-fold cross-validation, splitting the data presented in Section III-A into training (90%) and validation (10%) sets. For experiment (1), the data was augmented online using the configuration described in Section III-B. The network was trained for 1000 epochs. For each epoch, 2048 samples were generated for training and 256 for validation. The hyperparameters were selected based on the random search reported in Table II. The ConvNet evaluated in experiment (1) achieved a mean validation accuracy of 0.7914 (SD = 0.0287) over 10 folds. Figure 2 shows the learning curve for the best fold (accuracy: 0.8398).

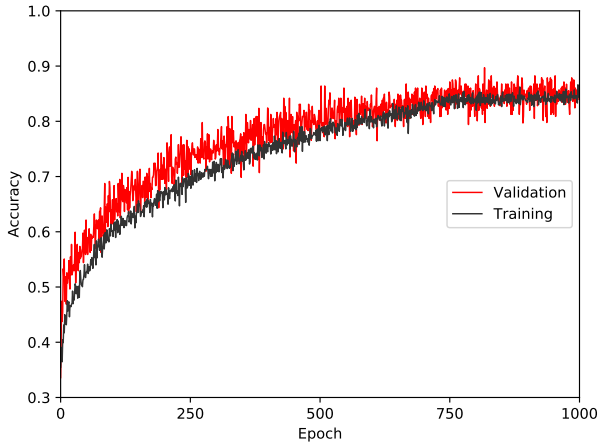

Fig. 2. Learning curve for the best fold of experiment (1): training a convolutional neural network (ConvNet) from scratch using data augmentation. Training on an NVIDIA Tesla K40 GPU, the duration of each epoch is approximately 12 seconds.

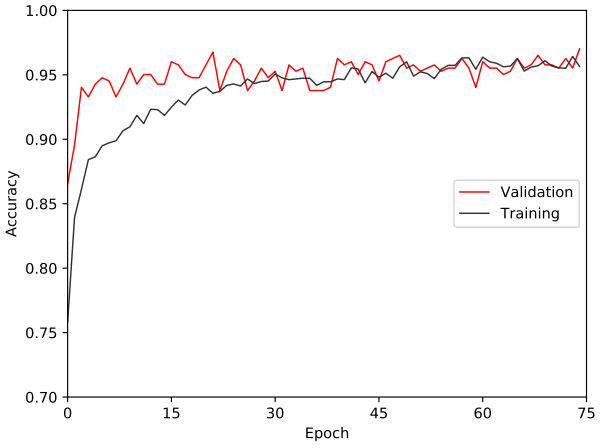

Fig. 3. Learning curve for the best fold of experiment (2): transfer learning using a residual neural network (ResNet50). Training on an NVIDIA Tesla K40 GPU, the duration of each epoch is less than a second.

Fig. 4. Examples from the hard testing set and their associated predictions using best folds for both networks. Note that the images have been resized into square format to illustrate the input to the neural networks. False predictions are marked in red.

For experiment (2), the entire data set described in Section III-A was fed to ResNet50 pre-trained on ImageNet for feature extraction, recording the output of the final average pooling layer to obtain a 2048-dimensional feature vector for each image. These features were then split into training and validation sets and fed to a fully-connected neural network with 512 nodes, training for 75 epochs. The hyperparameters used corresponded to those found during random search, as set out in Table II. The mean accuracy achieved over 10 folds was 0.9518 (SD = 0.0094). Figure 3 shows the learning curve for the best fold in experiment 2 (accuracy: 0.9701).

Figures 2 and 3 show that while both networks are able to learn to distinguish between the different classes, ConvNet reaches a plateau after 750 epochs. Despite data augmentation, the source data does not appear to be varied enough to improve ConvNet performance any further. Transfer learning using ResNet, in turn, shows that the model pre-trained on ImageNet has learned many of the features necessary for distinguishing between the different vehicle types. This results in high accuracy after only a few epochs, which may be improved by training further.

To assess the capability of the networks to generalize on data not included in the set described in Section III-A, both networks were also evaluated against a hard testing set containing 30 images from the conflict zone gathered from various sources. These images were intended to reflect the possible sources of content circulating in social media, ranging from high-quality professional press photographs to screen captures from news videos, and from user-generated videos to screenshots of social media postings and content.

Table III reports accuracy at various levels on the hard testing set for both the best fold and the mean over 10 folds. The 0.5 and 0.25 levels are included to reflect the possibility of using the tool to flag images for which the Softmax probability exceeds the given threshold. This allows, for instance, accounting for misclassifications between different vehicle types and images that contain multiple vehicle types. As Table III shows, ResNet significantly outperforms ConvNet at all levels of accuracy.

TABLE III
PERFORMANCE ON THE HARD TESTING SET

Architecture	Accuracy	Accuracy at 0.5	Accuracy at 0.25
(1) ConvNet			
Best	0.533	0.433	0.667
Mean	0.440	0.353	0.663
(2) ResNet			
Best	0.900	0.900	0.933
Mean	0.877	0.853	0.940

VI. CONCLUSIONS

This paper presented a system developed for recognizing military vehicles in social media images using deep learning. Built entirely using openly available deep learning libraries, architectures and pre-trained models, the paper investigated

two different solutions for training a vehicle recognition system with limited training data. The paper experimented with different neural network architectures: convolutional neural networks and residual neural networks.

The results showed that while both networks achieve a reasonable validation accuracy, residual neural networks are able to generalize much better when using transfer learning, that is, leveraging the features learned from some other distribution to describe the images in the current data set. Training a convolutional neural network from scratch using data augmentation to synthesize additional data could not match the level of performance achieved with transfer learning.

In addition to introducing additional vehicles to the system, future work could involve evaluating its performance using geographically-located, historical social media data pertaining to the Ukraine crisis, in order to assess how well the observations made by the tool correlate with the events as they unfolded. Moreover, as social media data is often multimodal, that is, the contents are presented using multiple modes of expression, such as photographs and written language, another possibility would be to process the image captions as well. Natural language processing techniques such as named-entity recognition could help to retrieve place names and other crucial metadata to contextualize the images.

Acknowledgments

The author would like to thank *CSC - IT Center for Science Ltd.* for computing resources, and Flickr users 131561895@N06 and 39259915@N02 for making the photographs used in Figure 1 available with a Creative Commons license. In addition, the author would like to thank the Emil Aaltonen Foundation (Tampere, Finland) for funding his travel and participation to the 2017 IEEE International Conference on Intelligence and Security Informatics in Beijing, China.

References

[1] T. Simon, A. Goldberg, L. Aharonson-Daniel, D. Leykin, and B. Adini, "Twitter in the cross fire – the use of social media in the Westgate Mall terror attack in Kenya," *PLoS ONE*, vol. 9, no. 8, p. e104136, 2014.

[2] M. Sienkiewicz, "Open BUK: Digital labor, media investigation and the downing of MH17," *Critical Studies in Media Communication*, vol. 32, no. 3, pp. 208–223, 2015.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.

[4] D. Omand, J. Bartlett, and C. Miller, "Introducing social media intelligence (SOCMINT)," *Intelligence and National Security*, vol. 27, no. 6, pp. 801–823, 2012.

[5] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems (NIPS'12)*, 2012, pp. 1097–1105.

[6] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *Journal of Big Data*, vol. 2, no. 1, pp. 1–21, 2015.

[7] K. L. O'Halloran, S. Tan, P. Wignell, J. A. Bateman, D. Pham, M. Grossman, and A. Vande Moere, "Interpreting text and image relations in violent extremist discourse: A mixed methods approach for big data analytics," *Terrorism and Political Violence*, pp. 1–21, 2016. [Online]. Available: http://dx.doi.org/10.1080/09546553.2016.1233871

[8] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," *arXiv*, vol. 1312.6229, 2013. [Online]. Available: http://arxiv.org/abs/1312.6229

[9] Z. Dong, M. Pei, Y. He, T. Liu, Y. Dong, and Y. Jia, "Vehicle type classification using unsupervised convolutional neural network," in *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR'14)*, 2014, pp. 172–177.

[10] H. Huttunen, F. S. Yancheshmeh, and K. Chen, "Car type recognition with deep neural networks," in *Proceedings of IEEE Intelligent Vehicles Symposium (IV'16)*, 2016, pp. 1115–1120.

[11] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPR'14)*, 2014, pp. 512–519.

[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA and London: MIT Press, 2017.

[14] F. Chollet, "Keras: Deep learning library for Theano and Tensorflow," https://github.com/fchollet/keras, 2015–.

[15] L. Shao, F. Zhu, and X. Li, "Transfer learning for visual categorization: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 5, pp. 1019–1034, 2015.

[16] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'16)*, 2016, pp. 770–778.

[18] ——, "Identity mappings in deep residual networks," in *Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, the Netherlands, Proceedings, Part IV*. Cham: Springer, 2016, pp. 630–645.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on Machine Learning (ICML'15)*, 2015, pp. 448–456.

[20] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML'10)*, 2010, pp. 807–814.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, 2009, pp. 248–255.

[22] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 281–305, 2012.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[24] The Theano Development Team, "Theano: a Python framework for fast computation of mathematical expressions," *CoRR*, vol. abs/1605.02688, 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[25] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. J. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Józefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. G. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. A. Tucker, V. Vanhoucke, V. Vasudevan, F. B. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous distributed systems," *arXiv*, vol. abs/1603.04467, 2016. [Online]. Available: http://arxiv.org/abs/1603.04467