Improving Deep Learning with Generic Data Augmentation

Luke Taylor
Department of Computer Science
University of Cape Town
Cape Town, South Africa
Email: tylchr001@myuct.ac.za

Geoff Nitschke
Department of Computer Science
University of Cape Town
Cape Town, South Africa
Email: gnitschke@cs.uct.ac.za

***Abstract*—Deep artificial neural networks require a large corpus of training data in order to effectively learn, where collection of such training data is often expensive and laborious. *Data augmentation* overcomes this issue by artificially inflating the training set with label preserving transformations. Recently there has been extensive use of generic data augmentation to improve *Convolutional Neural Network* (CNN) task performance. This study benchmarks various popular data augmentation schemes to allow researchers to make informed decisions as to which training methods are most appropriate for their data sets. Various geometric and photometric schemes are evaluated on a coarse-grained data set using a relatively simple CNN. Experimental results, run using 4-fold cross-validation and reported in terms of Top-1 and Top-5 accuracy, indicate that *cropping* in *geometric augmentation* significantly increases CNN task performance.**

I. Introduction

Convolutional Neural Networks (CNNs) [1], synonymous with deep learning are a hierarchical model of learning with multiple levels of representations, where higher levels capture more abstract concepts. A CNNs connectivity pattern, inspired by the animal visual cortex, enables the recognition and learning of spatial data such as images [1], audio [2] and text [3]. With recent developments of large data sets and increased computing power, CNNs have managed to achieve state-of-the-art results in various computer vision tasks including large scale image and video classification [4]. However, an issue is that most large data sets are not publicly available and training a CNN on small data-sets makes it prone to over-fitting, inhibiting the CNNs capability to generalize to unseen invariant data.

A potential solution is to use *Data Augmentation* (DA) [5], which is a regularization scheme that artificially inflates the data-set by using label preserving transformations to add more invariant examples. Generic DA is a set of computationally inexpensive methods [6], previously used to reduce over-fitting in training a CNN for the *ImageNet Large-Scale Visual Recognition Challenge* (ILSVRC) [7], and achieved state-of the-art results at the time. This augmentation scheme consists of *Geometric* and *Photometric* transformations [8], [9].

Geometric transformations alter the geometry of the image with the aim of making the CNN invariant to change in position and orientation. Example transformations include flipping, cropping, scaling and rotating. Photometric transformations amend the color channels with the objective of making the CNN invariant to change in lighting and color. For example, color jittering and *Fancy Principle Component Analysis (PCA)* [6], [10].

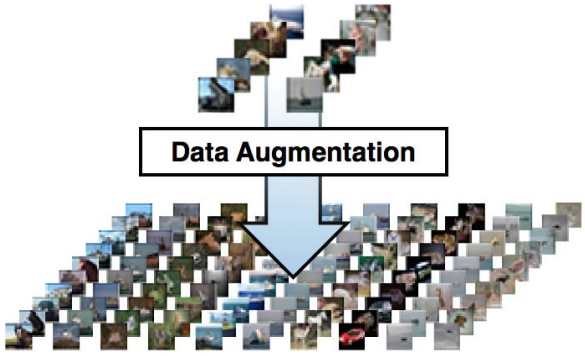

Fig. 1. *Data Augmentation* (DA) artificially inflates data-sets using label preserving transformations.

Complex DA is a scheme that artificially inflate the data set by using domain specific synthesization to produce more training data. This scheme has become increasingly popular [11], [12] as it has the ability to generate *richer* training data compared to the generic augmentation methods. For example, Masi et al. [13] developed a facial recognition system using synthesized faces with different poses and facial expressions to increase appearance variability enabling comparable task performance to state of the art facial recognition systems using less training data. At the frontier of data synthesis are *Generative Adversarial Networks* (GANs) [14] that have the ability to generate new samples after being trained on samples drawn from some distribution. For example, Zhang et al. [15] used a stack construction of GANs to generate realistic images of birds and flowers from text descriptions.

Thus, DA is a scheme to further boost CNN performance and prevent over-fitting. The use of DA is especially well-suited when the training data is limited or laborious to collect. Complex DA, albeit being a powerful augmentation scheme, is computationally expensive and time-consuming to implement. A viable option is to apply generic DA as it is computationally inexpensive and easy to implement.

Prevalent studies that comparatively evaluate various popular DA methods include those outlined in table I and described

978-1-5386-9276-9/18/$31.00

	Coarse-Grained	Geometric DA	Photometric DA
Chatfield *et al.*	✓	✓	
Mash *et al.*		✓	
Our benchmark	✓	✓	✓

TABLE I. PROPERTIES OF STUDIES THAT INVESTIGATE DA.

in the following. Chatfield et al. [16] addressed how different CNN architectures compared to each other by evaluating them on a common data-set. This study was mainly focused on rigorous evaluation of deep architectures and shallow encoding methods, though an evaluation of three augmentation methods was included. These consisted of flipping and combining flipping and cropping on training images in the coarse grained *Caltech 101*[1] and *Pascal VOC*[2] data-sets. In additional experiments the authors trained a CNN using gray-scale images, though lower task performance was observed. Overall results indicated that combining flipping and cropping yielded an increased task performance of $2 \sim 3\%$. A shortcoming of this study was the few DA methods evaluated.

Mash et al. [17] bench-marked a variety of geometric augmentation methods for the task of aircraft classification, using a fine-grained data-set of 10 classes. Augmentation methods tested included cropping, rotating, re-scaling, polygon occlusion and combinations of these methods. The cropping scheme combined with occlusion yielded the most benefits, achieving a 9% improvement over a benchmark task performance. Although this study evaluated various DA methods, photometric methods were not investigated and none were bench-marked on a coarse-grained data-set.

In line with this work, further research [18] noted that certain augmentation methods benefit from fine-grained data-sets. For example, extensive use of rotating training images to increase CNN task performance for galaxy morphology classification using a fine-grained data-set [18].

However, to date, there has been no comprehensive studies that comparatively evaluate various popular DA methods on large coarse-grained data-sets in order to ascertain the most appropriate DA method for any given data-set. Hence, this study's objective is to evaluate a variety of popular geometric and photometric augmentation schemes on the coarse grained Caltech101 data-set. Using a relatively simple CNN based on that used by Zeiler and Fergus [19], the goal is to contribute to empirical data in the field of deep-learning to enable researchers to select the most appropriate generic augmentation scheme for a given data-set.

II. DATA AUGMENTATION (DA) METHODS

DA refers to any method that artificially inflates the original training set with label preserving transformations and can be represented as the mapping:

$$\phi : \mathcal{S} \mapsto \mathcal{T}$$

Where, $\mathcal{S}$ is the original training set and $\mathcal{T}$ is the augmented set of $\mathcal{S}$. The artificially inflated training set is thus represented as:

$$\mathcal{S}' = \mathcal{S} \cup \mathcal{T}$$

Where, $\mathcal{S}'$ contains the original training set and the respective transformations defined by ϕ. Note the term label preserving transformations refers to the fact that if image x is an element of class y then $\phi(x)$ is also an element of class y.

As there is an endless array of mappings $\phi(x)$ that satisfy the constraint of being label preserving, this paper evaluates popular augmentation methods used in recent literature [16], [17] as well as a new augmentation method (section II-B). Specifically, seven augmentation methods were evaluated (figure 2), where one was a *No-Augmentation* method which acted as the task performance benchmark for all the experiments given three geometric and three photometric methods.

A. Geometric Methods

These are transformations that alter the geometry of the image by mapping the individual pixel values to new destinations. The underlying shape of the class represented within the image is preserved but altered to some new position and orientation. Given their success in related work [6], [18], [20] we investigated the *flipping*, *rotation* and *cropping* schemes.

Flipping mirrors the image across its vertical axis. It is one of the most used augmentation schemes after being popularized by Krizhevsky et al. [6]. It is computationally efficient and easy to implement due to only requiring rows of image matrices to be reversed.

The rotation scheme rotates the image around its center via mapping each pixel (x, y) of an image to (x', y') with the following transformation [18], [20]:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Where, exploratory experiments indicated that setting Θ as -30 degrees and $+30$ degrees establishes rotations that are large enough to generate new invariant samples.

Cropping is another augmentation scheme popularized by Krizhevsky et al. [6]. We used the same procedure as in related work [16], which consisted of extracting 224×224 crops from the four corners and the center of the 256×256 image.

B. Photometric Methods

These are transformations that alter the RGB channels by shifting each pixel value (r, g, b) to new pixel values (r', g', b') according to pre-defined heuristics. This adjusts image lighting and color and leaves the geometry unchanged. We investigated the *color jittering*, *edge enhancement* and *fancy PCA* photometric methods.

Color jittering is a method that either uses random color manipulation [21] or set color adjustment [20]. We selected the latter due to its accessible implementation using a *Hue*, *Saturation*, *Brightness* (HSB) filter[3].

[1] $www.vision.caltech.edu/ImageDatasets/Caltech101/$

[2] $host.robots.ox.ac.uk/pascal/VOC/$

[3] www.jhlabs.com/ip/filters/HSBAdjustFilter.html

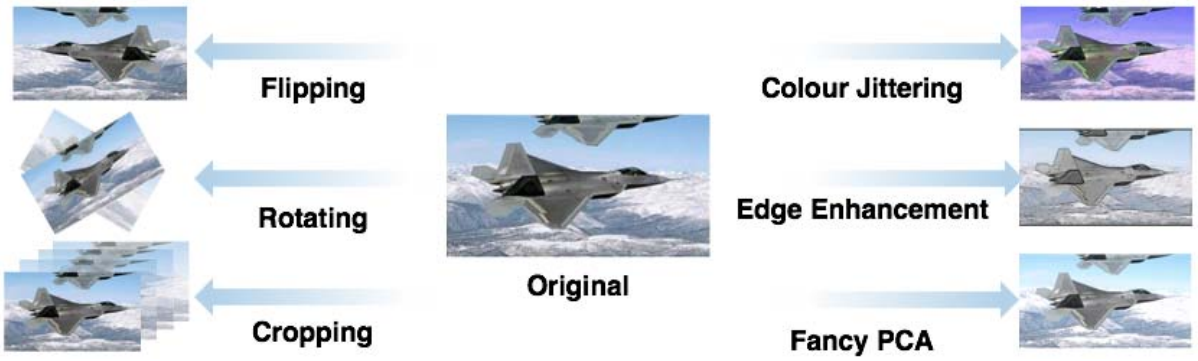

Fig. 2. Overview of the *Data Augmentation* (DA) methods evaluated.

Edge enhancement is a new augmentation scheme that enhances the contours of the class represented within the image. As the learned kernels in the CNN identify shapes it was hypothesized that CNN performance could be boosted by providing training images with intensified contours. This augmentation scheme was implemented as presented in *Algorithm 1*. Edge filtering was accomplished using the *Sobel* edge detector [22] where edges were identified by computing the local gradient $\nabla\mathcal{S}(i,j)$ at each pixel in the image $\mathcal{S}$.

ALGORITHM 1

Require: Source image: $\mathcal{I}$

$\mathcal{T} \leftarrow$ edge filter $\mathcal{I}$
$\mathcal{T} \leftarrow$ grayscale $\mathcal{T}$
$\mathcal{T} \leftarrow$ inverse $\mathcal{T}$
$\mathcal{I}' \leftarrow$ composite $\mathcal{T}$ over $\mathcal{I}$

return $\mathcal{I}'$

ALGORITHM 2

Require: Source image: $\mathcal{I}$

$\mathcal{M} \leftarrow$ Create a $255^2 \times 3$ matrix where the columns represent the RGB channels and all entries are the RGB values of $\mathcal{I}$. PCA is performed on $\mathcal{M}$.

for all Pixels $I(x,y)$ in I **do**

$[\mathcal{I}^R_{xy}, \mathcal{I}^G_{xy}, \mathcal{I}^B_{xy}]^T \leftarrow$ Add $\frac{1}{s_p}\mathcal{P}[\alpha_1\lambda_1, \alpha_2\lambda_2, \alpha_3\lambda_3]^T$

- $\mathcal{P}$ is a 3×3 matrix where the columns are the eigenvectors
- λ_i is the i$^{\text{th}}$ eigenvalue corresponding to the eigenvector $[p_{i,1}, p_{i,1}, p_{i,1}]^T$
- α_i is a random variable which is drawn from a Gaussian with 0 mean and standard deviation 0.1
- s_p is the scaling parameter which was initialised to $5 \cdot 10^6$ by trial and error.

end for
return $\mathcal{I}$

Fancy PCA is a scheme that performs PCA on sets of RGB pixels throughout the training set by adding multiples of principles components to the training images (Algorithm 2). In related work [6] it is unclear as to whether the authors performed PCA on individual images or on the entire training set. However, due to computation and memory constraints we adopted the former approach.

III. CNN Architecture

A CNN architecture was developed with the objective of obtaining a favorable tradeoff between task performance and training speed. Training speed was crucial as the CNN had to be trained seven times on data-sets ranging from ~ 8.5 to ~ 42.5 thousand images using $4-$fold cross validation. Also, the CNN had to be *deep* enough (containing enough parameters) so as the network would fit the training data.

We used an architecture (figure 3) similar to that described by [19], where exploratory experiments indicated a reasonable tradeoff between topology size and training speed. The architecture consisted of 5 trainable layers: 3 convolutional layers, 1 fully connected layer and 1 softmax layer. The CNN took a 3 channel (representing the RGB channels) 255×225 image as input, 30 filters of size 6×6 with a stride of 2 and a padding of 1 were convolved over the image producing a feature map of size $30 \times 110 \times 110$. This layer was compressed by a max-pooling filter of size 3×3 with stride of 2 which reduced it to a new feature map of dimensions $30 \times 55 \times 55$.

A set of 40 filters followed with the same size and stride as before which were convolved over the layer producing a new feature map of size $40 \times 27 \times 27$. Again a max pooling function of the same size and stride was applied which further reduced the feature map to $40 \times 13 \times 13$. This was fed into a last CNN layer using a set of 60 filters of size 3×3 of stride 1 producing a layer of $60 \times 11 \times 11$ parameters. Finally this was fed into a fully connected layer of size 140 which connected to a soft-max layer of size 101.

Overlapping pooling was deployed which increased CNN performance by reducing over-fitting [6]. This yielded sums of overlapping neighboring groups of neurons in the same feature map. A fully connected layer of 140 neurons was chosen as increased sizes did not generate greater improvements in performance [16]. Exploratory experiments indicated that smaller layer sizes resulted in richer encodings of the distinct classes

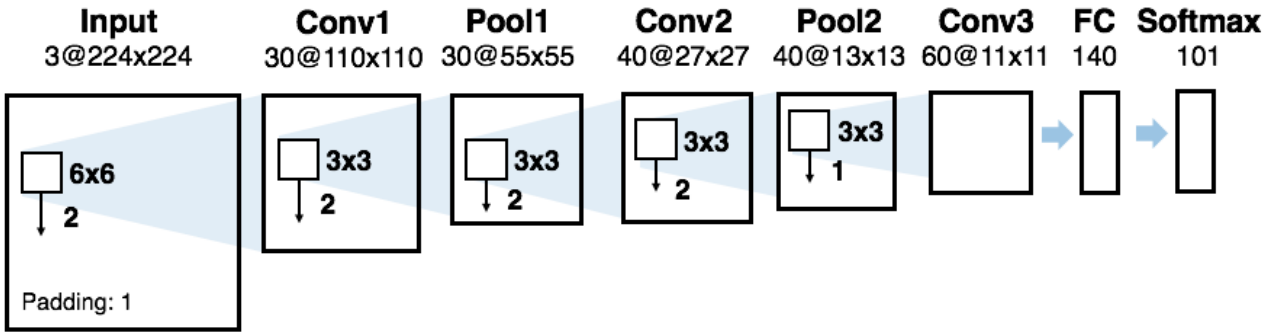

Fig. 3. The *Convolutional Neural Network* (CNN) architecture.

Hyper-parameter	Type
Activation Function	ReLu
Weight Initialisation	Xavier
Learning Rate	0.01
Optimisation Algorithm	SGD
Updater	Nesterov
Regularization	L2 Gradient Normalization
Minibatch	16
Epochs	30

TABLE II. HYPER-PARAMETERS USED BY THE CNN IN THIS STUDY.

yielding better generalization. The depth sizes of individual convolutional layers were determined by trial and error. Further convolutional layers did not increase performance considerably and were thus omitted to reduce computational time. It was also found that a second fully connected layer did not improve task performance.

All neurons used a *Rectified Linear Unit* [23] with the weights initially being initialised from a Gaussian distribution with a 0 mean and a standard deviation of 0.01. An initialisation scheme known as Xavier [24] was deployed which mitigated slow convergence to local minima. All weights were updated using back-propagation and stochastic gradient descent with a learning rate of 0.01. A variety of update functions were tested including *Adam* [25] and *Adagard* [26], however we selected *Nesterov* [27] with a momentum of 0.90 which we found to converge relatively fast and not suffer from from numerical instability and stagnating convergence.

Additionally, regularization was deployed in the form of gradient normalization with a *L2* of $5 \cdot 10^{-4}$ to reduce overfitting. Hyper-parameters for *activation function*, *learning rate*, *optimisation algorithm* and *updater* were based on those used in related work [6] and all other parameter values were determined by exploratory experiments. All CNN parameters used in this study are presented in table II.

IV. EXPERIMENTS

This study evaluated various DA methods on the Caltech101 data-set, which is a coarse-grained data-set consisting of 102 categories containing a total of 9144 images. The Caltech101 data-set was chosen as it is a well established data-set for training CNNs containing a large amount of varying classes [16], [19]. For CNN training most images in the Caltech101 data-set were used. That is, 725 images, including the *background* category in the Caltech101 data-set as it contained many uncorrelated images, were omitted. Also, further trimming was applied such that the cardinality of every class was divisible by 4 for cross-validation, which further increased the number images excluded, meaning that 8424 images in total were evaluated.

We elected to use cross-validation which maximized the use of selected images and better estimated how the CNNs performance would scale to other unknown data-sets. Specifically 4-fold[4] cross-validation was used which partitioned the data-set into 4 equal sized subsets, where 3 subsets were used for training and the other for validation purposes.

All images within the data-set were transformed to a size of 256×256, where every image was downsized such that the largest dimension was equal to 256. This downsized image was then centrally drawn on top of a 256×256 black image[5]. This enabled all augmentation schemes to have access to the full image in a fixed resolution of 256×256. Finally the transformed images underwent normalization by scaling all pixel values from $[0, 255] \rightarrow [0, 1]$.

Every CNN was trained using 30 epochs. This value was determined by exploratory experiments that evaluated validation and test scores every epoch. All implementation was completed in *Java 8* using *DL4J*[6] with the experiments being conducted on a *NVIDIA Tesla K80* GPU using *CUDA*. All source code and experiment details can be found online[7].

V. RESULTS AND DISCUSSION

Table III presents experimental results, where Top-1 and Top-5 scores were evaluated as percentages as done in the *Imagenet* competition [7], though we report accuracies rather than error rates. The CNN's output is a multinomial distribution over all classes:

$$\sum p_{class} = 1$$

[4]Higher fold cross validation would have taken too long to train.
[5]Numeric value of 0 for all channels thus acting as zero padding.
[6]*deeplearning4j.org*
[7]*github.com/webstorms/AugmentedDatasets*

	Top-1 Accuracy	Top-5 Accuracy
Baseline	48.13 ± 0.42%	64.50 ± 0.65%
Flipping	49.73 ± 1.13%	67.36 ± 1.38%
Rotating	50.80 ± 0.63%	69.41 ± 0.48%
Cropping	**61.95 ± 1.01%**	**79.10 ± 0.80%**
Color Jittering	**49.57 ± 0.53%**	67.18 ± 0.42%
Edge Enhancement	49.29 ± 1.16%	66.49 ± 0.84%
Fancy PCA	49.41 ± 0.84%	**67.54 ± 1.01%**

TABLE III. CNN TRAINING RESULTS FOR EACH DA METHOD.

The Top-1 score is the number of times the highest probability is associated with the correct target over all testing images. The Top-5 score is the number of times the correct label is contained within the 5 highest probabilities. As the CNN's accuracy was evaluated using cross-validation a standard deviation was associated with every result, thus indicating how variable the result is over different testing folds. In table III, Top-1 and Top-5 scores in the geometric and photometric DA category are represented in bold.

Results indicate that in all cases of applying DA, CNN classification task performance increased. Notably, the *geometric augmentation* schemes outperformed the *photometric schemes* in terms of Top-1 and Top-5 scores. The exception was the *flipping* scheme Top-5 score being inferior to the *fancy PCA* Top-5 score. For all schemes, a standard deviation of $0.5\% \sim 1\%$ indicated similar results over all folds with the cross-validation.

The cropping scheme yielded the greatest improvement in Top-1 score with an improvement of 13.82% in classification accuracy. Results also indicated that Top-5 classification yielded a similar task improvement which corroborated related work [16], [17]. We theorize that the *cropping* scheme outperforms the other methods as it generates more sample images than the other augmentation schemes. This increase in training data reduces the likelihood of over-fitting, improving generalization and thus increasing overall classification task performance. Also, cropping represents specific translations allowing the CNN exposure to a greater receptive view of training images which the other augmentation schemes do not take advantage of [16].

However, the *photometric* augmentation methods yielded modest improvements in performance compared to the geometric schemes, indicating the CNN yields increased task performance when trained on images containing invariance in geometry rather than lighting and color. The most appropriate photometric schemes were found to be *color jittering* with a top-1 classification improvement of 1.44% and *fancy PCA* which improved top-5 classification by 3.04%. *Fancy PCA* increased top-1 performance by 1.28% which supported the findings of previous work [6].

We also hypothesize that *color jittering* outperformed the other photometric schemes in top-1 classification as this scheme generated augmented images containing more variation compared to the other methods (figure 2). Also, *edge enhancement* augmentation did not yield comparable task performance, likely due to the overlay of the transformed image onto the source image (as described in section II-B) did not enhance the contours enough but rather lightened the entire image. However, the exact mechanisms responsible for the variability of CNN classification task performance given *geometric* versus *photometric* augmentation methods for coarse-grained data-sets remains the topic of ongoing research.

VI. CONCLUSIONS

This study's results demonstrate that an effective method of increasing CNN classification task performance is to make use of *Data Augmentation* (DA). Specifically, having evaluated a range of DA schemes using a relatively simple CNN architecture we were able to demonstrate that *geometric augmentation* methods outperform photometric methods when training on a coarse-grained data-set (that is, the Caltech101 data-set). The greatest task performance improvement was yielded by specific translations generated by the *cropping* method with a Top-1 score increase of 13.82%. These results indicate the importance of augmenting coarse-grained training data-sets using transformations that alter the geometry of the images rather than just lighting and color.

Future work will experiment with different coarse-grained data-sets to establish whether results obtained using the Caltech101 are transferable to other data-sets. Additionally different CNN architectures as well as other DA methods and combinations of DA methods will be investigated in comprehensive studies to ascertain the impact of applying a broad range of generic DA methods on coarse-grained data-sets.

REFERENCES

[1] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems*, 1996, pp. 396–404.

[2] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 2014, pp. 1533–1545.

[3] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in Neural Information Processing Systems*, 2014, pp. 2042–2050.

[4] O. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," in *British Machine Vision Conference*, 2015, p. 6.

[5] L. Yaeger, R. Lyon, and B. Webb, "Effective training of a neural network character classifier for word recognition," in *Advances in Neural Information Processing Systems*, 1996, pp. 807–813.

[6] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[7] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, and M. Bernstein, "Imagenet large scale visual recognition challenge," in *Advances in Neural Information Processing Systems*, 2016, pp. 3108–3116.

[8] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Berlin, Germany: Springer, 2014, pp. 346–361.

[9] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*. Montreal, Canada: NIPS Foundation, Inc., 2014, pp. 568–576.

[10] A. Howard, "Some improvements on deep convolutional neural network based image classification," in *arXiv preprint arXiv:1312.5402*, 2013.

[11] G. Rogez and C. Schmid, "MoCap-guided data augmentation for 3D pose estimation in the wild," in *International Journal of Computer Vision*, 2015, pp. 211–252.

[12] X. Peng, B. Sun, K. Ali, and K. Saenko, "Learning deep object detectors from 3d models," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1278–1286.

[13] I. Masi, A. Tran, T. Hassner, J. Leksut, and G. Medioni., "Do we really need to collect millions of faces for effective face recognition," in *European Conference on Computer Vision*. Springer International Publishing, 2016, pp. 579–596.

[14] 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.

[15] H. Zhang, T. Xu, H. Li, S. Zhang, X. Huang, X. Wang, and D. Metaxas, "Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks," in *arXiv preprint arXiv:1612.03242*, 2016.

[16] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the devil in the details: Delving deep into convolutional nets," in *British Machine Vision Conference*, 2014.

[17] R. Mash, B. Borghetti, and J. Pecarina, "Improved aircraft recognition for aerial refueling through data augmentation in convolutional neural networks," in *International Symposium on Visual Computing*. Springer, 2016, pp. 113–122.

[18] S. Dieleman, K. Willett, and J. Dambre, "Rotation-invariant convolutional neural networks for galaxy morphology prediction," in *Monthly Notices of the Royal Astronomical Society*, 2015, pp. 1441–1459.

[19] M. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*. Springer International Publishing, 2014, pp. 818–833.

[20] A. 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*, 2014, pp. 806–813.

[21] R. Wu, S. Yan, Y. Shan, Q. Dang, and G. Sun, "Deep image: Scaling up image recognition," in *arXiv preprint arXiv:1501.02876*, 2015.

[22] R. Gonzalez and R. Woods, "Digital image processing," in *Addison Wesley*, 1992, pp. 414–428.

[23] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and H. Seung, "Digital selection and analogue amplification coesist in a cortex-inspired silicon circuit," *Nature*, vol. 405, pp. 947–951, 2000.

[24] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Aistats, volume 9*, 2010, pp. 249–256.

[25] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *arXiv preprint arXiv:1412.6980*, 2014.

[26] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12(1), pp. 2121–2159, 2011.

[27] Y. Nesterov, "A method of solving a convex programming problem with convergence rate O(1/k2)," *Soviet Mathematics Doklady*, vol. 27(2), pp. 372–376, 1983.