Review of Literature

alikhanimjd

November 2020

Abstract

This literature review aims to cover machine learning engineering techniques discussed in the advanced machine learning course. Reliable seminal neural network architectures [24, 17, 56] are often reasonable convenient choices to solve a problem as opposed to experimenting a new model from scratch. In order to draw the most benefit from these models, one needs to be familiar with potential issues as well as case-dependent techniques to exploit and enhance the performance. Weight initialization, data augmentation, scheduling the learning rate, batch normalization, etc., are among the topics covered to address computation and data bottlenecks.

1 Introduction

With the staggering progress of deep learning, seminal architectures have become the go-to solution for many problems. However, the advancements are not limited to novel architectures. Research has also benefited from training and optimization techniques which often yield significant improvements. Batch Normalization, train and test time augmentation, drop out layers, weight decaying, weight initialization etc., are among common practice deep learning techniques applied to enhance the results if duly applied.

Data Augmentation is the de facto solution to data scarcity. Deep learning networks' success depends heavily on the size of data sets[50], however, for many problems such data is just not available. For instance in medical projects, patient's data privacy concerns restricts access large piles of data which is often sorely needed to develop a safe and accurate model [28]. Similarly, many small start-ups cannot afford the data or they take on a new problem for which the data sets are still scant. Moreover, introducing data with a starkly different distribution is not always favorable compared to taking advantage of domain knowledge to augment the already well curated data.

Batch normalization is used for standardization of a training batch's distribution [23] leading to stabilized and faster training.

Regularization methods refer to the set of techniques which aim at reducing model's capacity to mitigate or eliminate over-fitting [12]. Dropout layers, early stopping and weight decay are among popular examples of regularization.

different parameter initialization techniques and keeping track of the learning rate at which the parameters are updated have also proven to be helpful in achieving a better accuracy[12].

In order to aptly introduce a technique, aside from skill that comes from experience, one needs be familiar with the corresponding literature and theory. There exist a extensive body of research investigating the impact of each method and comparing the accuracy measurements with and without each one. This work attempts to provide a general overview of definitions and reported investigations.

2 Data Augmentation

It is a common knowledge that deep learning networks thrive in abundance of data. More data, even of not the best quality, can help the network given it comes from the same distribution[34]. High quality data is not always readily available, however, it is possible to compile a rather small carefully curated data-set to later augment it with the help of expert knowledge [55]. Data Augmentation can be deemed as a regularization method which particularly in image processing has demonstrated benefits [57]. This can be attributed to enhancement of intra-class invariance and inter-class distinctiveness. Intra-class invariance is referred to prediction of same class when a data point is replaced by another from the same class [41]. Inter-class distinctiveness is the model's ability to distinguish the changed class when the data point is replaced with data from another class. Typical data augmentation techniques aim for affine changes in color and geometry including cropping, rotation, scaling, blurring and several other image manipulation methods that do not change the structure of the image entirely. Early examples used to improve classification task on the MNIST data-set can be found in [5]. Non-affine transformations are not useless either. In handwritting recognition, miscellaneous styles can be represented by elastic distortions as shown in [9, 8, 45]. More advanced techniques include applications of Generative Adversarial Networks (GANs) [37] to modify an image by adding a theme or style to it [53]. Wang and Perez [3] explore the effectiveness of traditional transformation methods to double their data on MNIST [27] and tiny-imagenet and witness a 4 to 7 percent increase in the accuracy of the model. Howard uses three transformations to experience approximately 20% improvement on the visual recognition task of imagenet. The first transformation cropped the image from $256*256$ to $224*224$. The second was a horizontal flip and the third was to add randomly generated lightings. In [58] Wu expands ImageNet data-set with augmentations such as vignetting, rotation, flipping, cropping, color casting etc.

Random erasing is a technique that removes a patch of image randomly to reduce model's dependence on a particular feature in the input image. This trains a model more robust to occlusion. [59] shows about 0.5% improvement in accuracy using this technique on CIFAR-10 and CIFAR-100 datasets. Other problems may leave room for non-conventional augmentations as well. Pedestrian detection for self driving cars can gain better accuracy by adding synthetic occlusion or synthetic pedestrians to the background [33, 2].

[22] is show-case of augmentation in medical problems. Gaussian noise, flips, scaling, jittering, powers, Gaussian blur, rotations, etc., were among the augmentation techniques used. Rotation and Gaussian blur proved most effectiveness while noise diminished the results.

Data augmentation can be done in two ways. To have a better control over the proportions of each transformation and numbers of added data in general, one would be better off with off-line augmentation. On-line augmentation which happens during the training, can introduce a diversity as the model encounters unseen data on each iteration. The latter greatly fends off overfitting [9].

2.1 Test Time Augmentation (TTA)

Data augmentation is not only limited to training time. Research has shown that using augmentation during inference is advantageous as well [43]. The idea is the same, i.e. each image comes with transformed copies that help model make a better decision. Predictions on these copies are then pooled together applying various pooling methods to obtain a prediction. This provides the model with more than one chance for each image and intuitively would let the model see the image from different angles and perspectives (depending on the augmentations) to make a better decision. Besides improving accuracy [24, 52, 46, 29], test time augmentation helps to obtain more robustness [35, 48, 10] and estimates of uncertainty [47, 4].

Among methods used to improve accuracy, robustness, etc., such as tuning hyper parameters, test time augmentation really stands out in that it very easy to implement. There is no need to make any changes to the model and would be specially beneficial for smaller models. However, as there is no free lunch in machine learning, test time augmentation increases the inference time since for each predictions there are multiple runs through the network. Horizontal flips, cropping, brightness modifications and rotations are among the popular test time transformations.

Howard [20] explores test time augmentation among other methods to improve the model's performance. He arrives at a helpful set of transformations with a greedy approach, i.e. adding the augmentation if it increases the accuracy.

3 Batch Normalization

Since 2015 paper [23], batch normalization has been ubiquitously used in training deep learning networks [15, 44]. One of the reasons deep neural networks struggle in training could be attributed to changes in distribution from one mini batch to the next all while the weights are being updated. On each iteration, an estimate of error is used to determine the amount of change in parameters of each layer under the assumption that parameters of all the previous layers are constant [12]. Although the distribution of outputs from previous layers that are inputs to the current layer change, update of parameters is done assuming it does not [23]. As a consequence, optimizer will be chasing a constantly moving target. Batch normalization aims to salvage this problem by standardizing the output of each layer. By keeping the distribution of inputs to each layer almost constant, training is stabilized and speeds up. However, experiments have shown [40] that normalization smooths the function to be optimized in such a way that makes larger learning rates safe and therefore offers faster convergence.

In practice, two parameters γ and β are learned as the desired standard deviation and mean for output of each layer. restricting the mean and standard deviation to be zero and one respectively, limits the expressive power of neural network.

Specifying mean and standard deviation is not all it takes to control a distribution and the distribution will in fact change. Besides that, parameters γ and β would nudge mean and standard deviation from zero and one nevertheless. These two observations suggest that normalization of input is not the sole reason behind faster and stable training of a neural network. A Taylor expansion to approximate update in each step can shed some light on the inner workings of the model. Considering the second order for simplicity:

$$f(w) \approx f(w_0) + (w - w_0)^T g + \frac{1}{2}(w - w_0)^T H(w - w_0) \quad (1)$$

In which g and H stand for gradient and Hessian of the function f respectively. Replacing w with $w - \epsilon g$ in which ϵ controls the size of step taken in the update we have:

$$f(w - \epsilon g) = f(w_0) - \epsilon g^T g + \frac{1}{2}\epsilon^2 g^T H g \quad (2)$$

Zooming in on the term $\frac{1}{2}\epsilon^2 g^T H g$, if the function is linear close to constant it will amount to zero resulting in an ever decreasing loss. For complex functions which is usually what is dealt with, the term may be large enough to even increase the loss. Higher degree functions equates larger curvature blowing up the mentioned term as well as complicating the remaining terms in the approximation. It is not practical to control the update by decreasing ϵ as it may need to drop too small and slow down the training. Normalization on the other hand ensures a mean of zero and a standard deviation of one. This will suppress the magnitude of higher order interactions and allows for larger learning rates to be used.

4 Weight initialization

An important decision in training a neural network is where to start the training from. There are staggering number of parameters in a deep network the arbitrary initialization of which does not always work in our favor. Romero [39] mentions the difficulty of training deep networks with a uniform initialization. In fact, arbitrary initialization can lead to learning impairments [31]. One of the main issues with a bad initialization is the shrinkage of variance in inputs to the deeper layers in the network [26]. A small variance means virtually the same inputs over and over which stalls the learning. As another example, Initializing CNNs [24] with Gaussian noise setting mean to zero and standard deviation to one was a popular method which proved problematic in deeper networks. Overlooking the downstream activation functions while scaling the signals by k in each layer, can cause the blow up or drop of values by k^L resulting in very small gradients for a function like sigmoid.

On the contrary, a good initialization can expedite the training considerably. Saxe et al. [42] proposes orthonormal matrix initialization that outperforms Gassuian noise initialization. He et al. [15] uses a ReLU-aware initialization to observe this in VGGnet. Glorot et al [11] formulated a method to estimate a standard deviation based on number of input and output channels which He [16] later geared for ReLU non-linearity. Abbott and Sussillo [51] proposed an initialization that keeps the norm of backpropagated errors constant.

There are several options at our disposal to initialize the parameters depending on the nature of the problem and the network structure.

4.1 Zero-initialization

Bias parameters in networks are often initialized with zero. Initializing weights with zero causes all the outputs to be zero and subsequently, all the derivatives to have the same value. This practically reduces the model's capacity to a linear model.

4.2 Random Initialization

Random initialization is a common practice especially for baseline models to break the symmetry faced in zero-initialization. However as mentioned above, random weights can cause vanishing or exploding gradients [11].

4.3 Xavier initialization

Assigning values to the weights prior to knowledge about how the network would react to data cannot involve a very deterministic approach. However, there are ways to ward off issues such as vanishing or exploding gradients to some extent. A good way of initializing the weights is sampling from a Gaussian distribution with zero mean. Considering a linear function:

$$y = w_1x_1 + w_2x_2 + ... + w_nx_n + b \tag{3}$$

The variance of distribution needs to remain within a close proximity of the original variance to avoid vanishing or exploding. This is a clue to the initialization for which Xavier et al. [11] propose a solution.

To keep the variance the same, let us compute the new variance from 3:

$$var(y) = var(w_1x_1 + w_2x_2 + ...w_nx_n) \tag{4}$$

and

$$var(w_ix_i) = E(x_i)^2var(w_i)+E(w_i)^2var(x_i)+var(w_i)var(x_i) \tag{5}$$

Considering E to be zero (zero mean for the Gaussian distribution) equation 5 simplifies to:

$$var(w_ix_i) = var(w_i)var(x_i) \tag{6}$$

replacing the result in 4:

$$var(y) = N \times var(w_i) \times var(x_i) \tag{7}$$

therefore to keep the variance constant we need to have $var(w_i) = 1/N$.

This Xavier initialization is a popular implementation available in many deep learning frameworks.

4.4 He initialization

Xavier initialization was driven under the assumption of having linear activation function which is a reasonable approximation of sigmoid or tanh functions for normalized output with zero mean. With the introduction of ReLU to mitigate vanishing and exploding gradient, Xavier initialization needed some adjustment as it did not turn quite compatible.

$$Var(w_l)\& = \frac{2}{n_l}$$

$$SD(w_l)\& = \sqrt{\frac{2}{n_l}}$$

He et al. propose this initialization to observe significant performance gains in ImageNet classification [16].

4.5 LSUV initialization

One of the downsides to He initialization is the exclusivity to ReLU function. Although ReLU is a very common choice for an activation function, that will be a problem in more complex architectures. In addition, it turns out the mean of a layer is usually closer to 0.5 than zero and standard deviation is not exactly equal to 1. Very deep networks will face issues as a result.

In order to avoid customizing He initialization complex architectures, LSUV initialization can be used [31] to keep the variance of output in check. The weight matrix for each layer is initialized with an orthogonal initialization. Then, for each layer a minibatch input is fed to divide the weights by the deviation of the output. This will continue in a loop until all output's variance fall within a proximity of 1. Alo et al. use this initialization scheme to achieve their best results for their novel network IRCNN [1].

5 Regularization

One of the notorious problems in training a neural network is over-fitting [6]. When the networks is ac accustomed to examples from training set too much, would have a very low training error while not doing great on the validation set indicated by larger validation error. Figure **??** shows an example of training and validation loss for an over-fit model.

It can be seen that the network is ... after no... iteration. Depending on the situation, there are a few remedies.

Early stopping A simple solution to curb the difference between validation and training error would be to terminate training before earlier; iteration number ... for example in figure. Different criteria can be used for early stopping such as difference between validation and training error, the amount of validation error drop on each iteration, etc. Prechelt [36] proposed 3 ways in this regard.

In general, there are options to control the model's capacity to avoid over-fitting: 1- Change in the structure of the network. 2- Change in the values of weights [32]. Techniques such as dropout or reducing the number nodes in a layer fall under the former while L^2 regularization [54] are examples of the latter.

Weight decay Small weights and parameters restrain the capacity of the model by making it more robust to fluctuations while larger weights project small changes in input to very large changes in output [42]. It is more common to refer to techniques that reduce the weights of the model as regularization methods. Weight decay is one of the simplest regularization methods.

Weight decay is performed by penalizing the magnitude of weights as another term in the loss function expression. Equation 8 shows a common instance of weight decay.

$$Loss = MSE(y^{\wedge}, y) + w_d \times sum(w^2) \tag{8}$$

Where w_d is a hyper-parameter which determines the extent to which the weights are penalized. Both L1 and L2 weight decay have been used in the literature [14]. [25] theoretically explains the effect of weight decay in a simple setting. Merity et al. [30] use weight decay to optimize their LSTM language model. Hssayeni et al. use this method on popular AlexNet, ResNet and VGG models to detect distraction in drivers [21].

Dropout A simple way to restrain the model structurally is to cut out an arbitrary number of nodes with a certain probability [49]. The probability is a hyper parameter and higher values tend to remove more nodes from the network. The famous AlexNet [24] achieved state of the art performance at the time using dropout.

Weight constraint A substitute for weight decay would be to actually ensure that weights are within a boundary instead of encouraging them to be small. On each iteration all weights are reviewed to clip the ones over the boundary. Hinton used this technique to improve the classifier on ImageNet [18]. Srivastava et al. [49] used a weight constraint on their MNIST classifier. In another work [7] max norm weight constraint was used to avoid overfitting in an attention-based model.

Noise Research has also shown benefits of adding small amounts of noise to assist the model in generalization [38]. Adding noise in effect increases the size of input and intuitively can be conceived as an augmentation method. This help the model not to rely on the distribution of training set too much. One of the early cases of noise addition was done by Holmstrom [19] for training a Multilayer perceptron. Graves et al. improve generalization by adding noise to weights [13].

References

[1] Md Zahangir Alom et al. "Inception recurrent convolutional neural network for object recognition". In: *arXiv preprint arXiv:1704.07709* (2017).

[2] Saleh Aly et al. "Partially occluded pedestrian classification using part-based classifiers and Restricted Boltzmann Machine model". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE. 2013, pp. 1065–1070.

[3] Andrea Asperti and Claudio Mastronardo. "The effectiveness of data augmentation for detection of gastrointestinal diseases from endoscopical images". In: *arXiv preprint arXiv:1712.03689* (2017).

[4] Murat Seckin Ayhan and Philipp Berens. "Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks". In: (2018).

[5] HS Baird. "Document image analysis. chapter Document image defect models". In: *IEEE Computer Society Press, Los Alamitos, CA, USA* 2 (1995), pp. 315–325.

[6] Rich Caruana, Steve Lawrence, and C Lee Giles. "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping". In: *Advances in neural information processing systems*. 2001, pp. 402–408.

[7] Jan K Chorowski et al. "Attention-based models for speech recognition". In: *Advances in neural information processing systems*. 2015, pp. 577–585.

[8] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[9] Dan Claudiu Cireşan et al. “Deep, big, simple neural nets for handwritten digit recognition”. In: *Neural computation* 22.12 (2010), pp. 3207–3220.

[10] Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. “Certified adversarial robustness via randomized smoothing”. In: *arXiv preprint arXiv:1902.02918* (2019).

[11] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics.* 2010, pp. 249–256.

[12] Ian Goodfellow et al. *Deep learning.* Vol. 1. 2. MIT press Cambridge, 2016.

[13] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. “Speech recognition with deep recurrent neural networks”. In: *2013 IEEE international conference on acoustics, speech and signal processing.* IEEE. 2013, pp. 6645–6649.

[14] Song Han et al. “Learning both weights and connections for efficient neural network”. In: *Advances in neural information processing systems.* 2015, pp. 1135–1143.

[15] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[16] Kaiming He et al. “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: *Proceedings of the IEEE international conference on computer vision.* 2015, pp. 1026–1034.

[17] Kaiming He et al. “Identity mappings in deep residual networks”. In: *European conference on computer vision.* Springer. 2016, pp. 630–645.

[18] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012).

[19] Lasse Holmstrom and Petri Koistinen. “Using additive noise in back-propagation training”. In: *IEEE transactions on neural networks* 3.1 (1992), pp. 24–38.

[20] Andrew G Howard. “Some improvements on deep convolutional neural network based image classification”. In: *arXiv preprint arXiv:1312.5402* (2013).

[21] Murtadha D Hssayeni et al. “Distracted driver detection: Deep learning vs handcrafted features”. In: *Electronic Imaging* 2017.10 (2017), pp. 20–26.

[22] Zeshan Hussain et al. “Differential data augmentation techniques for medical imaging classification tasks”. In: *AMIA Annual Symposium Proceedings.* Vol. 2017. American Medical Informatics Association. 2017, p. 979.

[23] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[24] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[25] Anders Krogh and John A Hertz. “A simple weight decay can improve generalization”. In: *Advances in neural information processing systems.* 1992, pp. 950–957.

[26] Siddharth Krishna Kumar. “On weight initialization in deep neural networks”. In: *arXiv preprint arXiv:1704.08863* (2017).

[27] Yann LeCun, Corinna Cortes, and Christopher JC Burges. “The MNIST database of handwritten digits, 1998”. In: *URL http://yann. lecun. com/exdb/mnist* 10.34 (1998), p. 14.

[28] Ali Madani et al. “Semi-supervised learning with generative adversarial networks for chest x-ray classification with ability of data domain adaptation”. In: *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018).* IEEE. 2018, pp. 1038–1042.

[29] Kazuhisa Matsunaga et al. “Image classification of melanoma, nevus and seborrheic keratosis by deep neural network ensemble”. In: *arXiv preprint arXiv:1703.03108* (2017).

[30] Stephen Merity, Nitish Shirish Keskar, and Richard Socher. “Regularizing and optimizing LSTM language models”. In: *arXiv preprint arXiv:1708.02182* (2017).

[31] Dmytro Mishkin and Jiri Matas. “All you need is a good init”. In: *arXiv preprint arXiv:1511.06422* (2015).

[32] Grégoire Montavon et al. "Explaining nonlinear classification decisions with deep taylor decomposition". In: *Pattern Recognition* 65 (2017), pp. 211–222.

[33] Jonas Nilsson et al. "Pedestrian detection using augmented training data". In: *2014 22nd International Conference on Pattern Recognition*. IEEE. 2014, pp. 4548–4553.

[34] Luis Perez and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning". In: *arXiv preprint arXiv:1712.04621* (2017).

[35] Aaditya Prakash et al. "Deflecting adversarial attacks with pixel deflection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8571–8580.

[36] Lutz Prechelt. "Automatic early stopping using cross validation: quantifying the criteria". In: *Neural Networks* 11.4 (1998), pp. 761–767.

[37] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[38] Russell Reed and Robert J MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.

[39] Adriana Romero et al. "Fitnets: Hints for thin deep nets". In: *arXiv preprint arXiv:1412.6550* (2014).

[40] Shibani Santurkar et al. "How does batch normalization help optimization?" In: *Advances in Neural Information Processing Systems*. 2018, pp. 2483–2493.

[41] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. "Apac: Augmented pattern classification with neural networks". In: *arXiv preprint arXiv:1505.03229* (2015).

[42] Andrew M Saxe, James L McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *arXiv preprint arXiv:1312.6120* (2013).

[43] Jan Schlüter and Thomas Grill. "Exploring Data Augmentation for Improved Singing Voice Detection with Neural Networks." In: *ISMIR*. 2015, pp. 121–126.

[44] David Silver et al. "Mastering the game of go without human knowledge". In: *nature* 550.7676 (2017), pp. 354–359.

[45] Patrice Y Simard, David Steinkraus, John C Platt, et al. "Best practices for convolutional neural networks applied to visual document analysis." In: *Icdar*. Vol. 3. 2003. 2003.

[46] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[47] Lewis Smith and Yarin Gal. "Understanding measures of uncertainty for adversarial example detection". In: *arXiv preprint arXiv:1803.08533* (2018).

[48] Yang Song et al. "Pixeldefend: Leveraging generative models to understand and defend against adversarial examples". In: *arXiv preprint arXiv:1710.10766* (2017).

[49] Nitish Srivastava et al. "Dropout: a simple way to prevent neural networks from overfitting". In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[50] Chen Sun et al. "Revisiting unreasonable effectiveness of data in deep learning era". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 843–852.

[51] David Sussillo and LF Abbott. "Random walk initialization for training very deep feedforward networks". In: *arXiv preprint arXiv:1412.6558* (2014).

[52] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[53] Fabio Henrique Kiyoiti dos Santos Tanaka and Claus Aranha. "Data augmentation using GANs". In: *arXiv preprint arXiv:1904.09135* (2019).

[54] Twan Van Laarhoven. "L2 regularization versus batch and weight normalization". In: *arXiv preprint arXiv:1706.05350* (2017).

[55] Cristina Nader Vasconcelos and Bárbara Nader Vasconcelos. "Increasing deep learning melanoma classification by classical and expert knowledge based image transforms". In: *CoRR, abs/1702.07025* 1 (2017).

[56] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[57] Sebastien C Wong et al. "Understanding data augmentation for classification: when to warp?" In: *2016 international conference on digital image computing: techniques and applications (DICTA)*. IEEE. 2016, pp. 1–6.

[58] Ren Wu et al. "Deep image: Scaling up image recognition". In: *arXiv preprint arXiv:1501.02876* 7.8 (2015).

[59] Zhun Zhong et al. "Camera style adaptation for person re-identification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5157–5166.