Shorten and Khoshgoftaar *J Big Data* (2019) 6:60
https://doi.org/10.1186/s40537-019-0197-0

Journal of Big Data

SURVEY PAPER Open Access

A survey on Image Data Augmentation for Deep Learning

Connor Shorten* and Taghi M. Khoshgoftaar

*Correspondence:
cshorten2015@fau.edu
Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, Boca Raton, USA

Abstract

Deep convolutional neural networks have performed remarkably well on many Computer Vision tasks. However, these networks are heavily reliant on big data to avoid overfitting. Overfitting refers to the phenomenon when a network learns a function with very high variance such as to perfectly model the training data. Unfortunately, many application domains do not have access to big data, such as medical image analysis. This survey focuses on Data Augmentation, a data-space solution to the problem of limited data. Data Augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better Deep Learning models can be built using them. The image augmentation algorithms discussed in this survey include geometric transformations, color space augmentations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, generative adversarial networks, neural style transfer, and meta-learning. The application of augmentation methods based on GANs are heavily covered in this survey. In addition to augmentation techniques, this paper will briefly discuss other characteristics of Data Augmentation such as test-time augmentation, resolution impact, final dataset size, and curriculum learning. This survey will present existing methods for Data Augmentation, promising developments, and meta-level decisions for implementing Data Augmentation. Readers will understand how Data Augmentation can improve the performance of their models and expand limited datasets to take advantage of the capabilities of big data.

Keywords: Data Augmentation, Big data, Image data, Deep Learning, GANs

Introduction

Deep Learning models have made incredible progress in discriminative tasks. This has been fueled by the advancement of deep network architectures, powerful computation, and access to big data. Deep neural networks have been successfully applied to Computer Vision tasks such as image classification, object detection, and image segmentation thanks to the development of convolutional neural networks (CNNs). These neural networks utilize parameterized, sparsely connected kernels which preserve the spatial characteristics of images. Convolutional layers sequentially downsample the spatial resolution of images while expanding the depth of their feature maps. This series of convolutional transformations can create much lower-dimensional and more useful representations of images than what could possibly be hand-crafted. The success of CNNs has spiked interest and optimism in applying Deep Learning to Computer Vision tasks.

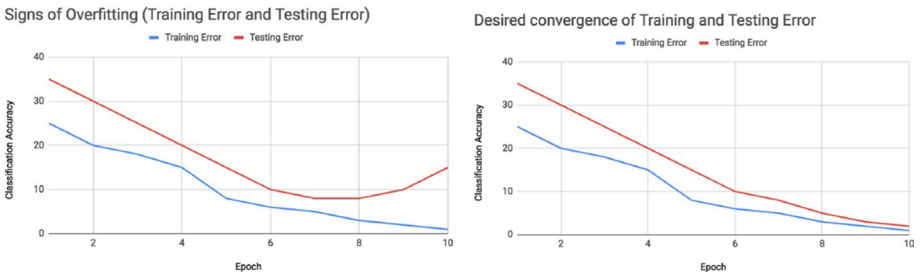

Fig. 1 The plot on the left shows an inflection point where the validation error starts to increase as the training rate continues to decrease. The increased training has caused the model to overfit to the training data and perform poorly on the testing set relative to the training set. In contrast, the plot on the right shows a model with the desired relationship between training and testing error

There are many branches of study that hope to improve current benchmarks by applying deep convolutional networks to Computer Vision tasks. Improving the generalization ability of these models is one of the most difficult challenges. Generalizability refers to the performance difference of a model when evaluated on previously seen data (training data) versus data it has never seen before (testing data). Models with poor generalizability have overfitted the training data. One way to discover overfitting is to plot the training and validation accuracy at each epoch during training. The graph below depicts what overfitting might look like when visualizing these accuracies over training epochs (Fig. 1).

To build useful Deep Learning models, the validation error must continue to decrease with the training error. Data Augmentation is a very powerful method of achieving this. The augmented data will represent a more comprehensive set of possible data points, thus minimizing the distance between the training and validation set, as well as any future testing sets.

Data Augmentation, the focus of this survey, is not the only technique that has been developed to reduce overfitting. The following few paragraphs will introduce other solutions available to avoid overfitting in Deep Learning models. This listing is intended to give readers a broader understanding of the context of Data Augmentation.

Many other strategies for increasing generalization performance focus on the model's architecture itself. This has led to a sequence of progressively more complex architectures from AlexNet [1] to VGG-16 [2], ResNet [3], Inception-V3 [4], and DenseNet [5]. Functional solutions such as dropout regularization, batch normalization, transfer learning, and pretraining have been developed to try to extend Deep Learning for application on smaller datasets. A brief description of these overfitting solutions is provided below. A complete survey of regularization methods in Deep Learning has been compiled by Kukacka et al. [6]. Knowledge of these overfitting solutions will inform readers about other existing tools, thus framing the high-level context of Data Augmentation and Deep Learning.

- Dropout [7] is a regularization technique that zeros out the activation values of randomly chosen neurons during training. This constraint forces the network to learn more robust features rather than relying on the predictive capability of a small subset of neurons in the network. Tompson et al. [8] extended this idea to convolutional

networks with Spatial Dropout, which drops out entire feature maps rather than individual neurons.

- Batch normalization [9] is another regularization technique that normalizes the set of activations in a layer. Normalization works by subtracting the batch mean from each activation and dividing by the batch standard deviation. This normalization technique, along with standardization, is a standard technique in the preprocessing of pixel values.
- Transfer Learning [10, 11] is another interesting paradigm to prevent overfitting. Transfer Learning works by training a network on a big dataset such as ImageNet [12] and then using those weights as the initial weights in a new classification task. Typically, just the weights in convolutional layers are copied, rather than the entire network including fully-connected layers. This is very effective since many image datasets share low-level spatial characteristics that are better learned with big data. Understanding the relationship between transferred data domains is an ongoing research task [13]. Yosinski et al. [14] find that transferability is negatively affected primarily by the specialization of higher layer neurons and difficulties with splitting co-adapted neurons.
- Pretraining [15] is conceptually very similar to transfer learning. In Pretraining, the network architecture is defined and then trained on a big dataset such as ImageNet [12]. This differs from Transfer Learning because in Transfer Learning, the network architecture such as VGG-16 [2] or ResNet [3] must be transferred as well as the weights. Pretraining enables the initialization of weights using big datasets, while still enabling flexibility in network architecture design.
- One-shot and Zero-shot learning [16, 17] algorithms represent another paradigm for building models with extremely limited data. One-shot learning is commonly used in facial recognition applications [18]. An approach to one-shot learning is the use of siamese networks [19] that learn a distance function such that image classification is possible even if the network has only been trained on one or a few instances. Another very popular approach to one-shot learning is the use of memory-augmented networks [20]. Zero-shot learning is a more extreme paradigm in which a network uses input and output vector embeddings such as Word2Vec [21] or GloVe [22] to classify images based on descriptive attributes.

In contrast to the techniques mentioned above, Data Augmentation approaches overfitting from the root of the problem, the training dataset. This is done under the assumption that more information can be extracted from the original dataset through augmentations. These augmentations artificially inflate the training dataset size by either data warping or oversampling. Data warping augmentations transform existing images such that their label is preserved. This encompasses augmentations such as geometric and color transformations, random erasing, adversarial training, and neural style transfer. Oversampling augmentations create synthetic instances and add them to the training set. This includes mixing images, feature space augmentations, and generative adversarial networks (GANs). Oversampling and Data Warping augmentations do not form a mutually exclusive dichotomy. For example, GAN samples can be stacked with random cropping to further inflate the dataset. Decisions around final dataset size,

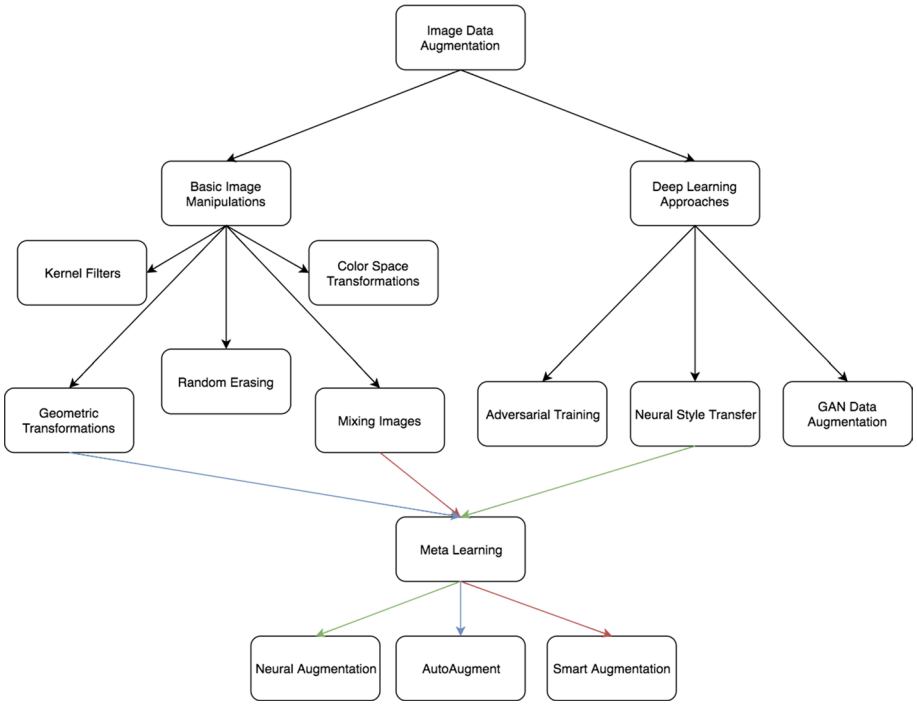

Fig. 2 A taxonomy of image data augmentations covered; the colored lines in the figure depict which data augmentation method the corresponding meta-learning scheme uses, for example, meta-learning using Neural Style Transfer is covered in neural augmentation [36]

test-time augmentation, curriculum learning, and the impact of resolution are covered in this survey under the "Design considerations for image Data Augmentation" section. Descriptions of individual augmentation techniques will be enumerated in the "Image Data Augmentation techniques" section. A quick taxonomy of the Data Augmentations is depicted below in Fig. 2.

Before discussing image augmentation techniques, it is useful to frame the context of the problem and consider what makes image recognition such a difficult task in the first place. In classic discriminative examples such as cat versus dog, the image recognition software must overcome issues of viewpoint, lighting, occlusion, background, scale, and more. The task of Data Augmentation is to bake these translational invariances into the dataset such that the resulting models will perform well despite these challenges.

It is a generally accepted notion that bigger datasets result in better Deep Learning models [23, 24]. However, assembling enormous datasets can be a very daunting task due to the manual effort of collecting and labeling data. Limited datasets is an especially prevalent challenge in medical image analysis. Given big data, deep convolutional networks have been shown to be very powerful for medical image analysis tasks such as skin lesion classification as demonstrated by Esteva et al. [25]. This has inspired the use of CNNs on medical image analysis tasks [26] such as liver lesion classification, brain scan analysis, continued research in skin lesion classification, and more. Many of the images studied are derived from computerized tomography (CT) and magnetic resonance imaging (MRI) scans, both of which are expensive and labor-intensive to collect. It is especially difficult to build big medical image datasets due to the rarity of diseases, patient

privacy, the requirement of medical experts for labeling, and the expense and manual effort needed to conduct medical imaging processes. These obstacles have led to many studies on image Data Augmentation, especially GAN-based oversampling, from the application perspective of medical image classification.

Many studies on the effectiveness of Data Augmentation utilize popular academic image datasets to benchmark results. These datasets include MNIST hand written digit recognition, CIFAR-10/100, ImageNet, tiny-imagenet-200, SVHN (street view house numbers), Caltech-101/256, MIT places, MIT-Adobe 5K dataset, Pascal VOC, and Stanford Cars. The datasets most frequently discussed are CIFAR-10, CIFAR-100, and ImageNet. The expansion of open-source datasets has given researchers a wide variety of cases to compare performance results of Data Augmentation techniques. Most of these datasets such as ImageNet would be classified as big data. Many experiments constrain themselves to a subset of the dataset to simulate limited data problems.

In addition to our focus on limited datasets, we will also consider the problem of class imbalance and how Data Augmentation can be a useful oversampling solution. Class imbalance describes a dataset with a skewed ratio of majority to minority samples. Leevy et al. [27] describe many of the existing solutions to high-class imbalance across data types. Our survey will show how class-balancing oversampling in image data can be done with Data Augmentation.

Many aspects of Deep Learning and neural network models draw comparisons with human intelligence. For example, a human intelligence anecdote of transfer learning is illustrated in learning music. If two people are trying to learn how to play the guitar, and one already knows how to play the piano, it seems likely that the piano-player will learn to play the guitar faster. Analogous to learning music, a model that can classify ImageNet images will likely perform better on CIFAR-10 images than a model with random weights.

Data Augmentation is similar to imagination or dreaming. Humans imagine different scenarios based on experience. Imagination helps us gain a better understanding of our world. Data Augmentation methods such as GANs and Neural Style Transfer can 'imagine' alterations to images such that they have a better understanding of them. The remainder of the paper is organized as follows: A brief "Background" is provided to give readers a historical context of Data Augmentation and Deep Learning. "Image Data Augmentation techniques" discusses each image augmentation technique in detail along with experimental results. "Design considerations for image Data Augmentation" discusses additional characteristics of augmentation such as test-time augmentation and the impact of image resolution. The paper concludes with a "Discussion" of the presented material, areas of "Future work", and "Conclusion".

Background

Image augmentation in the form of data warping can be found in LeNet-5 [28]. This was one of the first applications of CNNs on handwritten digit classification. Data augmentation has also been investigated in oversampling applications. Oversampling is a technique used to re-sample imbalanced class distributions such that the model is not overly biased towards labeling instances as the majority class type. Random Oversampling (ROS) is a naive approach which duplicates images randomly from the

minority class until a desired class ratio is achieved. Intelligent oversampling techniques date back to SMOTE (Synthetic Minority Over-sampling Technique), which was developed by Chawla et al. [29]. SMOTE and the extension of Borderline-SMOTE [30] create new instances by interpolating new points from existing instances via k-Nearest Neighbors. The primary focus of this technique was to alleviate problems due to class imbalance, and SMOTE was primarily used for tabular and vector data.

The AlexNet CNN architecture developed by Krizhevsky et al. [1] revolutionized image classification by applying convolutional networks to the ImageNet dataset. Data Augmentation is used in their experiments to increase the dataset size by a magnitude of 2048. This is done by randomly cropping 224×224 patches from the original images, flipping them horizontally, and changing the intensity of the RGB channels using PCA color augmentation. This Data Augmentation helped reduce overfitting when training a deep neural network. The authors claim that their augmentations reduced the error rate of the model by over 1%.

Since then, GANs were introduced in 2014 [31], Neural Style Transfer [32] in 2015, and Neural Architecture Search (NAS) [33] in 2017. Various works on GAN extensions such as DCGANs, CycleGANs and Progressively-Growing GANs [34] were published in 2015, 2017, and 2017, respectively. Neural Style Transfer was sped up with the development of Perceptual Losses by Johnson et al. [35] in 2016. Applying meta-learning concepts from NAS to Data Augmentation has become increasingly popular with works such as Neural Augmentation [36], Smart Augmentation [37], and AutoAugment [38] published in 2017, 2017, and 2018, respectively.

Applying Deep Learning to medical imaging has been a popular application for CNNs since they became so popular in 2012. Deep Learning and medical imaging became increasingly popular with the demonstration of dermatologist-level skin cancer detection by Esteva et al. [25] in 2017.

The use of GANs in medical imaging is well documented in a survey by Yi et al. [39]. This survey covers the use of GANs in reconstruction such as CT denoising [40], accelerated magnetic resonance imaging [41], PET denoising [42], and the application of super-resolution GANs in retinal vasculature segmentation [43]. Additionally, Yi et al. [39] cover the use of GAN image synthesis in medical imaging applications such as brain MRI synthesis [44, 45], lung cancer diagnosis [46], high-resolution skin lesion synthesis [47], and chest x-ray abnormality classification [48]. GAN-based image synthesis Data Augmentation was used by Frid-Adar et al. [49] in 2018 for liver lesion classification. This improved classification performance from 78.6% sensitivity and 88.4% specificity using classic augmentations to 85.7% sensitivity and 92.4% specificity using GAN-based Data Augmentation.

Most of the augmentations covered focus on improving Image Recognition models. Image Recognition is when a model predicts an output label such as 'dog' or 'cat' given an input image.

However, it is possible to extend results from image recognition to other Computer Vision tasks such as Object Detection led by the algorithms YOLO [50], R-CNN [51], fast R-CNN [52], and faster R-CNN [53] or Semantic Segmentation [54] including algorithms such as U-Net [55].

Image Data Augmentation techniques

The earliest demonstrations showing the effectiveness of Data Augmentations come from simple transformations such as horizontal flipping, color space augmentations, and random cropping. These transformations encode many of the invariances discussed earlier that present challenges to image recognition tasks. The augmentations listed in this survey are geometric transformations, color space transformations, kernel filters, mixing images, random erasing, feature space augmentation, adversarial training, GAN-based augmentation, neural style transfer, and meta-learning schemes. This section will explain how each augmentation algorithm works, report experimental results, and discuss disadvantages of the augmentation technique.

Data Augmentations based on basic image manipulations

Geometric transformations

This section describes different augmentations based on geometric transformations and many other image processing functions. The class of augmentations discussed below could be characterized by their ease of implementation. Understanding these transformations will provide a useful base for further investigation into Data Augmentation techniques.

We will also describe the different geometric augmentations in the context of their 'safety' of application. The safety of a Data Augmentation method refers to its likelihood of preserving the label post-transformation. For example, rotations and flips are generally safe on ImageNet challenges such as cat versus dog, but not safe for digit recognition tasks such as 6 versus 9. A non-label preserving transformation could potentially strengthen the model's ability to output a response indicating that it is not confident about its prediction. However, achieving this would require refined labels [56] post-augmentation. If the label of the image after a non-label preserving transformation is something like [0.5 0.5], the model could learn more robust confidence predictions. However, constructing refined labels for every non-safe Data Augmentation is a computationally expensive process.

Due to the challenge of constructing refined labels for post-augmented data, it is important to consider the 'safety' of an augmentation. This is somewhat domain dependent, providing a challenge for developing generalizable augmentation policies, (see AutoAugment [38] for further exploration into finding generalizable augmentations). There is no image processing function that cannot result in a label changing transformation at some distortion magnitude. This demonstrates the data-specific design of augmentations and the challenge of developing generalizable augmentation policies. This is an important consideration with respect to the geometric augmentations listed below.

Flipping

Horizontal axis flipping is much more common than flipping the vertical axis. This augmentation is one of the easiest to implement and has proven useful on datasets such as CIFAR-10 and ImageNet. On datasets involving text recognition such as MNIST or SVHN, this is not a label-preserving transformation.

Color space

Digital image data is usually encoded as a tensor of the dimension (height × width × color channels). Performing augmentations in the color channels space is another strategy that is very practical to implement. Very simple color augmentations include isolating a single color channel such as R, G, or B. An image can be quickly converted into its representation in one color channel by isolating that matrix and adding 2 zero matrices from the other color channels. Additionally, the RGB values can be easily manipulated with simple matrix operations to increase or decrease the brightness of the image. More advanced color augmentations come from deriving a color histogram describing the image. Changing the intensity values in these histograms results in lighting alterations such as what is used in photo editing applications.

Cropping

Cropping images can be used as a practical processing step for image data with mixed height and width dimensions by cropping a central patch of each image. Additionally, random cropping can also be used to provide an effect very similar to translations. The contrast between random cropping and translations is that cropping will reduce the size of the input such as (256,256) → (224, 224), whereas translations preserve the spatial dimensions of the image. Depending on the reduction threshold chosen for cropping, this might not be a label-preserving transformation.

Rotation

Rotation augmentations are done by rotating the image right or left on an axis between 1° and 359°. The safety of rotation augmentations is heavily determined by the rotation degree parameter. Slight rotations such as between 1 and 20 or −1 to −20 could be useful on digit recognition tasks such as MNIST, but as the rotation degree increases, the label of the data is no longer preserved post-transformation.

Translation

Shifting images left, right, up, or down can be a very useful transformation to avoid positional bias in the data. For example, if all the images in a dataset are centered, which is common in face recognition datasets, this would require the model to be tested on perfectly centered images as well. As the original image is translated in a direction, the remaining space can be filled with either a constant value such as 0 s or 255 s, or it can be filled with random or Gaussian noise. This padding preserves the spatial dimensions of the image post-augmentation.

Noise injection

Noise injection consists of injecting a matrix of random values usually drawn from a Gaussian distribution. Noise injection is tested by Moreno-Barea et al. [57] on nine datasets from the UCI repository [58]. Adding noise to images can help CNNs learn more robust features.

Geometric transformations are very good solutions for positional biases present in the training data. There are many potential sources of bias that could separate the

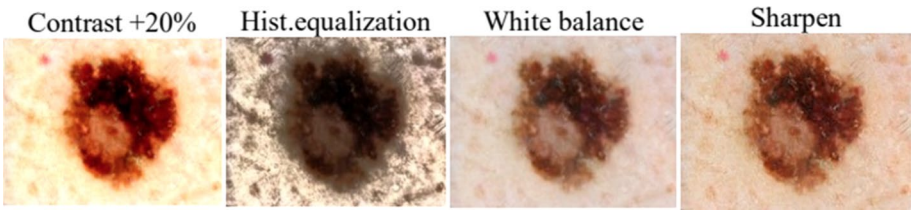

Fig. 3 Examples of Color Augmentations provided by Mikolajczyk and Grochowski [72] in the domain of melanoma classification

distribution of the training data from the testing data. If positional biases are present, such as in a facial recognition dataset where every face is perfectly centered in the frame, geometric transformations are a great solution. In addition to their powerful ability to overcome positional biases, geometric transformations are also useful because they are easily implemented. There are many imaging processing libraries that make operations such as horizontal flipping and rotation painless to get started with. Some of the disadvantages of geometric transformations include additional memory, transformation compute costs, and additional training time. Some geometric transformations such as translation or random cropping must be manually observed to make sure they have not altered the label of the image. Finally, in many of the application domains covered such as medical image analysis, the biases distancing the training data from the testing data are more complex than positional and translational variances. Therefore, the scope of where and when geometric transformations can be applied is relatively limited.

Color space transformations

Image data is encoded into 3 stacked matrices, each of size height × width. These matrices represent pixel values for an individual RGB color value. Lighting biases are amongst the most frequently occurring challenges to image recognition problems. Therefore, the effectiveness of color space transformations, also known as photometric transformations, is fairly intuitive to conceptualize. A quick fix to overly bright or dark images is to loop through the images and decrease or increase the pixel values by a constant value. Another quick color space manipulation is to splice out individual RGB color matrices. Another transformation consists of restricting pixel values to a certain min or max value. The intrinsic representation of color in digital images lends itself to many strategies of augmentation.

Color space transformations can also be derived from image-editing apps. An image's pixel values in each RGB color channel is aggregated to form a color histogram. This histogram can be manipulated to apply filters that change the color space characteristics of an image.

There is a lot of freedom for creativity with color space augmentations. Altering the color distribution of images can be a great solution to lighting challenges faced by testing data (Figs. 3, 4).

Image datasets can be simplified in representation by converting the RGB matrices into a single grayscale image. This results in smaller images, height × width × 1,

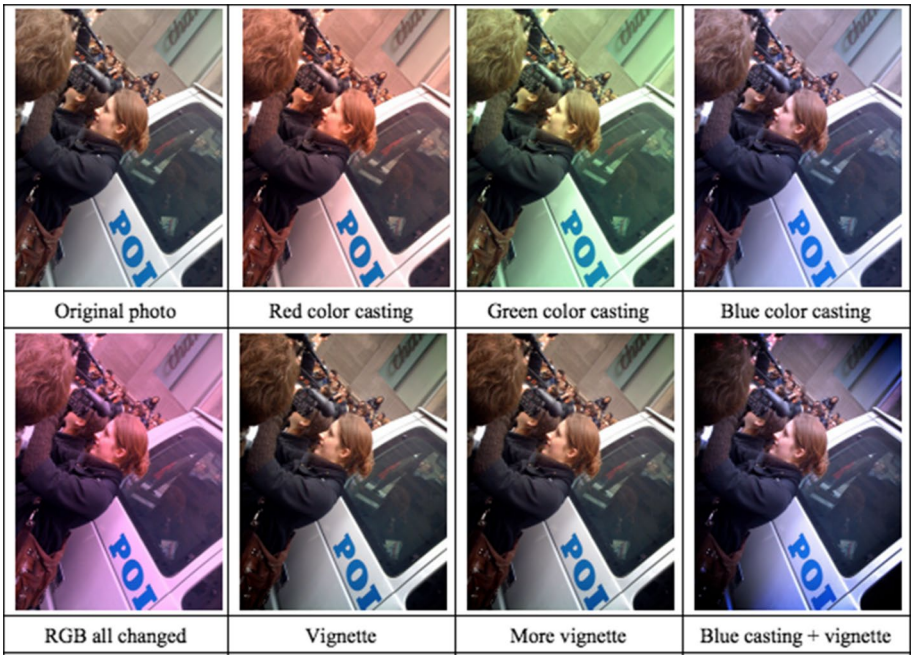

Fig. 4 Examples of color augmentations tested by Wu et al. [127]

resulting in faster computation. However, this has been shown to reduce performance accuracy. Chatifled et al. [59] found a ~3% classification accuracy drop between grayscale and RGB images with their experiments on ImageNet [12] and the PASCAL [60] VOC dataset. In addition to RGB versus grayscale images, there are many other ways of representing digital color such as HSV (Hue, Saturation, and Value). Jurio et al. [61] explore the performance of Image Segmentation on many different color space representations from RGB to YUV, CMY, and HSV.

Similar to geometric transformations, a disadvantage of color space transformations is increased memory, transformation costs, and training time. Additionally, color transformations may discard important color information and thus are not always a label-preserving transformation. For example, when decreasing the pixel values of an image to simulate a darker environment, it may become impossible to see the objects in the image. Another indirect example of non-label preserving color transformations is in Image Sentiment Analysis [62]. In this application, CNNs try to visually predict the sentiment score of an image such as: highly negative, negative, neutral, positive, or highly positive. One indicator of a negative/highly negative image is the presence of blood. The dark red color of blood is a key component to distinguish blood from water or paint. If color space transforms repeatedly change the color space such that the model cannot recognize red blood from green paint, the model will perform poorly on Image Sentiment Analysis. In effect, color space transformations will eliminate color biases present in the dataset in favor of spatial characteristics. However, for some tasks, color is a very important distinctive feature.

Table 1 Results of Taylor and Nitschke's Data Augmentation experiments on Caltech101 [63]

	Top-1 accuracy (%)	Top-5 accuracy (%)
Baseline	48.13 ± 0.42	64.50 ± 0.65
Flipping	49.73 ± 1.13	67.36 ± 138
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*

Their results find that the cropping geometric transformation results in the most accurate classifier

The italic value denote high performance according to the comparative metrics

Geometric versus photometric transformations

Taylor and Nitschke [63] provide a comparative study on the effectiveness of geometric and photometric (color space) transformations. The geometric transformations studied were flipping, −30° to 30° rotations, and cropping. The color space transformations studied were color jittering, (random color manipulation), edge enhancement, and PCA. They tested these augmentations with 4-fold cross-validation on the Caltech101 dataset filtered to 8421 images of size 256×256 (Table 1).

Kernel filters

Kernel filters are a very popular technique in image processing to sharpen and blur images. These filters work by sliding an $n \times n$ matrix across an image with either a Gaussian blur filter, which will result in a blurrier image, or a high contrast vertical or horizontal edge filter which will result in a sharper image along edges. Intuitively, blurring images for Data Augmentation could lead to higher resistance to motion blur during testing. Additionally, sharpening images for Data Augmentation could result in encapsulating more details about objects of interest.

Sharpening and blurring are some of the classical ways of applying kernel filters to images. Kang et al. [64] experiment with a unique kernel filter that randomly swaps the pixel values in an $n \times n$ sliding window. They call this augmentation technique PatchShuffle Regularization. Experimenting across different filter sizes and probabilities of shuffling the pixels at each step, they demonstrate the effectiveness of this by achieving a 5.66% error rate on CIFAR-10 compared to an error rate of 6.33% achieved without the use of PatchShuffle Regularization. The hyperparameter settings that achieved this consisted of 2×2 filters and a 0.05 probability of swapping. These experiments were done using the ResNet [3] CNN architecture (Figs. 5, 6).

Kernel filters are a relatively unexplored area for Data Augmentation. A disadvantage of this technique is that it is very similar to the internal mechanisms of CNNs. CNNs have parametric kernels that learn the optimal way to represent images layer-by-layer. For example, something like PatchShuffle Regularization could be implemented with a convolution layer. This could be achieved by modifying the standard convolution layer parameters such that the padding parameters preserve spatial resolution and the subsequent activation layer keeps pixel values between 0 and 255, in contrast to something like a sigmoid activation which maps pixels to values between 0 and 1. Therefore kernel

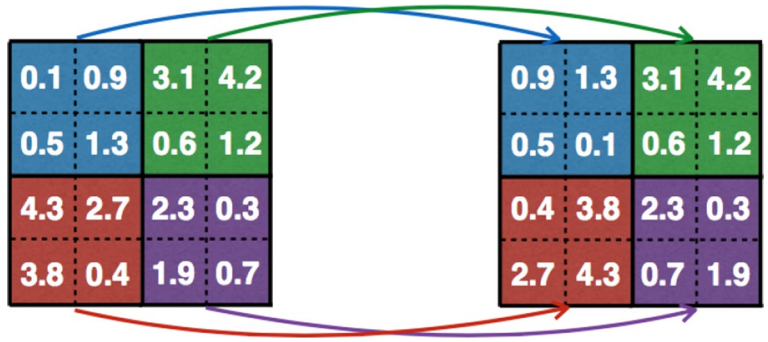

Fig. 5 Examples of applying the PatchShuffle regularization technique [64]

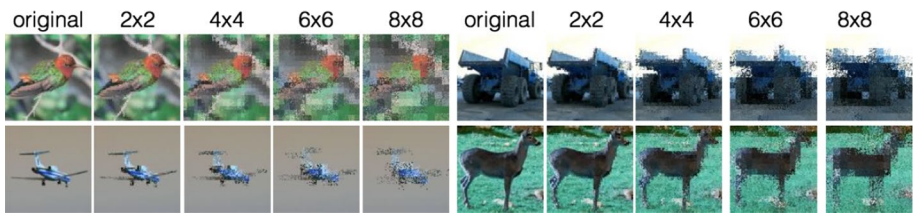

Fig. 6 Pixels in a n × n window are randomly shifted with a probability parameter p

filters can be better implemented as a layer of the network rather than as an addition to the dataset through Data Augmentation.

Mixing images

Mixing images together by averaging their pixel values is a very counterintuitive approach to Data Augmentation. The images produced by doing this will not look like a useful transformation to a human observer. However, Ionue [65] demonstrated how the pairing of samples could be developed into an effective augmentation strategy. In this experiment, two images are randomly cropped from 256×256 to 224×224 and randomly flipped horizontally. These images are then mixed by averaging the pixel values for each of the RGB channels. This results in a mixed image which is used to train a classification model. The label assigned to the new image is the same as the first randomly selected image (Fig. 7).

On the CIFAR-10 dataset, Ionue reported a reduction in error rate from 8.22 to 6.93% when using the SamplePairing Data Augmentation technique. The researcher found even better results when testing a reduced size dataset, reducing CIFAR-10 to 1000 total samples with 100 in each class. With the reduced size dataset, SamplePairing resulted in an error rate reduction from 43.1 to 31.0%. The reduced CIFAR-10 results demonstrate the usefulness of the SamplePairing technique in limited data applications (Fig. 8).

Another detail found in the study is that better results were obtained when mixing images from the entire training set rather than from instances exclusively belonging to the same class. Starting from a training set of size N, SamplePairing produces a dataset of size $N^2 + N$. In addition, Sample Pairing can be stacked on top of other augmentation techniques. For example, if using the augmentations demonstrated in

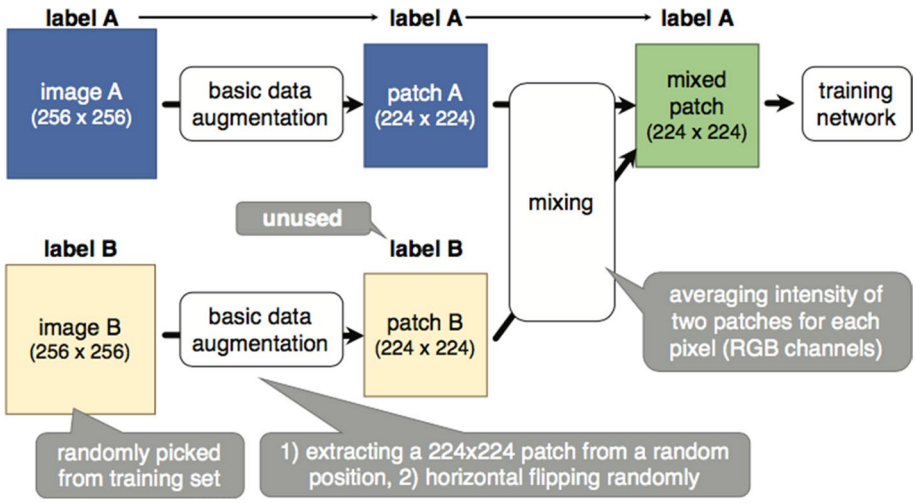

Fig. 7 SamplePairing augmentation strategy [65]

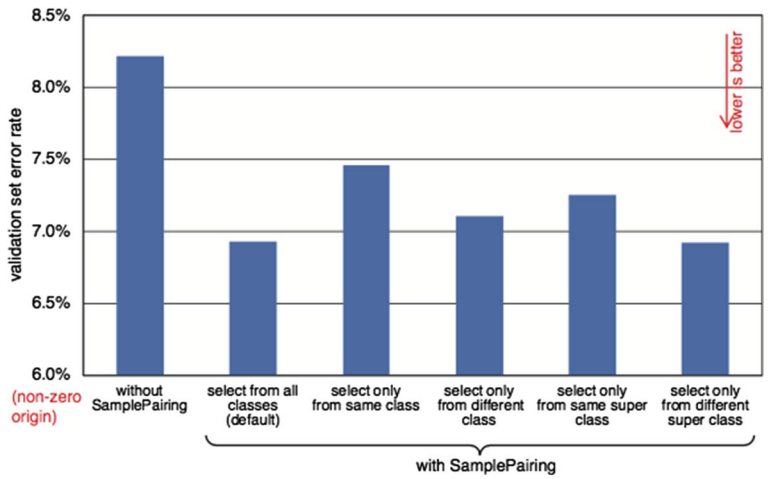

Fig. 8 Results on the reduced CIFAR-10 dataset. Experimental results demonstrated with respect to sampling pools for image mixing [65]

the AlexNet paper by Krizhevsky et al. [1], the 2048 × dataset increase can be further expanded to $(2048 \times N)^2$.

The concept of mixing images in an unintuitive way was further investigated by Summers and Dinneen [66]. They looked at using non-linear methods to combine images into new training instances. All of the methods they used resulted in better performance compared to the baseline models (Fig. 9).

Amongst these non-linear augmentations tested, the best technique resulted in a reduction from 5.4 to 3.8% error on CIFAR-10 and 23.6% to 19.7% on CIFAR-100. In like manner, Liang et al. [67] used GANs to produce mixed images. They found that the inclusion of mixed images in the training data reduced training time and increased the diversity of GAN-samples. Takahashi and Matsubara [68] experiment

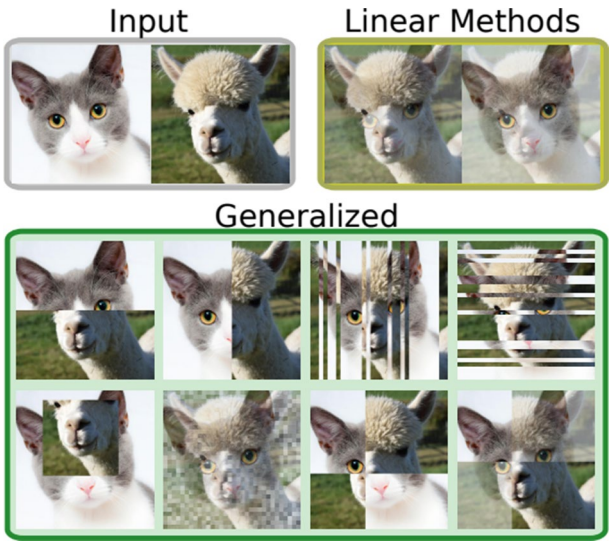

Fig. 9 Non-linearly mixing images [66]

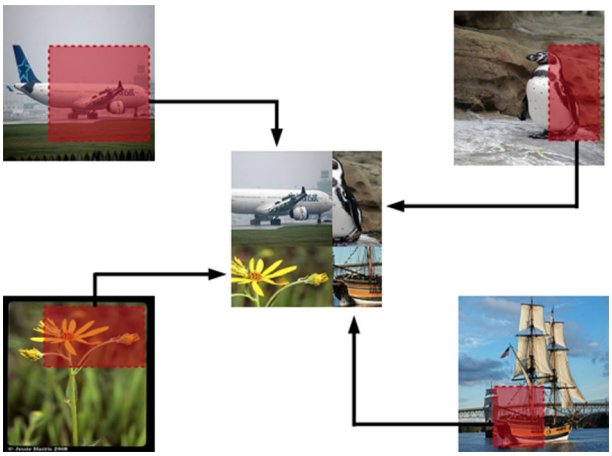

Fig. 10 Mixing images through random image cropping and patching [68]

with another approach to mixing images that randomly crops images and concatenates the croppings together to form new images as depicted below. The results of their technique, as well as SamplePairing and mixup augmentation, demonstrate the sometimes unreasonable effectiveness of big data with Deep Learning models (Fig. 10).

An obvious disadvantage of this technique is that it makes little sense from a human perspective. The performance boost found from mixing images is very difficult to understand or explain. One possible explanation for this is that the increased dataset size results in more robust representations of low-level characteristics such as lines and edges. Testing the performance of this in comparisons to transfer learning and pretraining methods is an interesting area for future work. Transfer learning and pretraining are other techniques that learn low-level characteristics in CNNs. Additionally, it will be

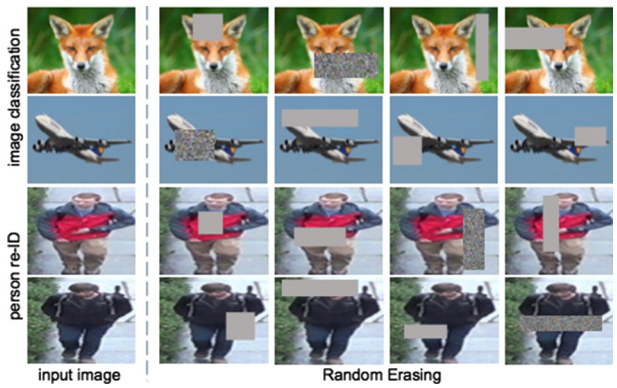

Fig. 11 Example of random erasing on image recognition tasks [70]

interesting to see how the performance changes if we partition the training data such that the first 100 epochs are trained with original and mixed images and the last 50 with original images only. These kinds of strategies are discussed further in Design Considerations of Data Augmentation with respect to curriculum learning [69]. Additionally, the paper will cover a meta-learning technique developed by Lemley et al. [37] that uses a neural network to learn an optimal mixing of images.

Random erasing

Random erasing [70] is another interesting Data Augmentation technique developed by Zhong et al. Inspired by the mechanisms of dropout regularization, random erasing can be seen as analogous to dropout except in the input data space rather than embedded into the network architecture. This technique was specifically designed to combat image recognition challenges due to occlusion. Occlusion refers to when some parts of the object are unclear. Random erasing will stop this by forcing the model to learn more descriptive features about an image, preventing it from overfitting to a certain visual feature in the image. Aside from the visual challenge of occlusion, in particular, random erasing is a promising technique to guarantee a network pays attention to the entire image, rather than just a subset of it.

Random erasing works by randomly selecting an $n \times m$ patch of an image and masking it with either 0 s, 255 s, mean pixel values, or random values. On the CIFAR-10 dataset this resulted in an error rate reduction from 5.17 to 4.31%. The best patch fill method was found to be random values. The fill method and size of the masks are the only parameters that need to be hand-designed during implementation (Figs. 11, 12).

Random erasing is a Data Augmentation method that seeks to directly prevent overfitting by altering the input space. By removing certain input patches, the model is forced to find other descriptive characteristics. This augmentation method can also be stacked on top of other augmentation techniques such as horizontal flipping or color filters. Random erasing produced one of the highest accuracies on the CIFAR-10 dataset. DeVries and Taylor [71] conducted a similar study called Cutout Regularization. Like the random erasing study, they experimented with randomly masking regions of the image (Table 2).

Mikolajcyzk and Grochowski [72] presented an interesting idea to combine random erasing with GANs designed for image inpainting. Image inpainting describes the task of

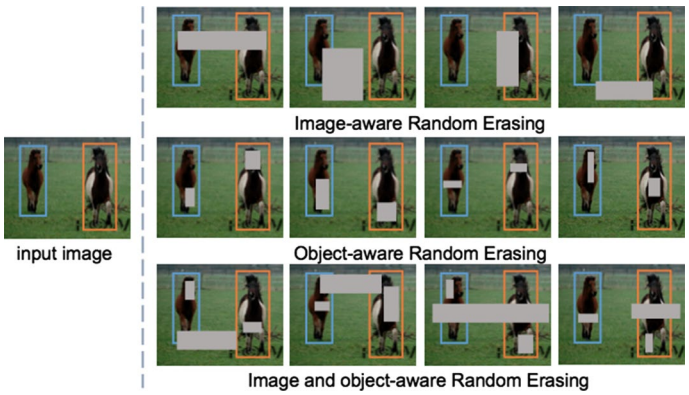

Fig. 12 Example of random erasing on object detection tasks [70]

Table 2 Results of Cutout Regularization [104], plus denotes using traditional augmentation methods, horizontal flipping and cropping

Method	C10	C10+	C100	C100+	SVHN
ResNetl8 [5]	10.63 ± 0.26	4.72 ± 0.21	36.68 ± 0.57	22.46 ± 0.31	–
ResNet18 + cutout	9.31 ± 0.18	3.99 ± 0.13	34.98 ± 0.29	21.96 ± 0.24	–
WideResNet [21]	6.97 ± 0.22	3.87 ± 0.08	26.06 ± 0.22	18.8 ± 0.08	1.60 ± 0.05
WideResNet + cutout	*5.54 ± 0.08*	3.08 ± 0.16	*23.94 ± 0.15*	18.41 ± 0.27	*1.30 ± 0.03*
Shake-shake regularization [4]	–	2.86	–	15.85	–
Shake-shake regularization + cutout	–	*2.56 ± 0.07*	–	*15.20 ± 0.21*	–

A 2.56% error rate is obtained on CIFAR-10 using cutout and traditional augmentation methods

The italic value denote high performance according to the comparative metrics

filling in a missing piece of an image. Using a diverse collection of GAN inpainters, the random erasing augmentation could seed very interesting extrapolations. It will be interesting to see if better results can be achieved by erasing different shaped patches such as circles rather than $n \times m$ rectangles. An extension of this will be to parameterize the geometries of random erased patches and learn an optimal erasing configuration.

A disadvantage to random erasing is that it will not always be a label-preserving transformation. In handwritten digit recognition, if the top part of an '8' is randomly cropped out, it is not any different from a '6'. In many fine-grained tasks such as the Stanford Cars dataset [73], randomly erasing sections of the image (logo, etc.) may make the car brand unrecognizable. Therefore, some manual intervention may be necessary depending on the dataset and task.

A note on combining augmentations

Of the augmentations discussed, geometric transformations, color space transformations, kernel filters, mixing images, and random erasing, nearly all of these transformations come with an associated distortion magnitude parameter as well. This parameter encodes the distortional difference between a 45° rotation and a 30° rotation. With a large list of potential augmentations and a mostly continuous space of magnitudes, it is easy to conceptualize the enormous size of the augmentation search space. Combining augmentations such as cropping, flipping, color shifts, and random erasing can result in

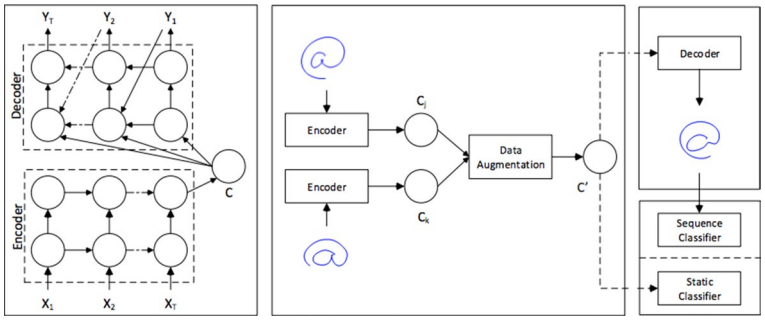

Fig. 13 Architecture diagram of the feature space augmentation framework presented by DeVries and Taylor [75]

Fig. 14 Examples of interpolated instances in the feature space on the handwritten '@' character [75]

massively inflated dataset sizes. However, this is not guaranteed to be advantageous. In domains with very limited data, this could result in further overfitting. Therefore, it is important to consider search algorithms for deriving an optimal subset of augmented data to train Deep Learning models with. More on this topic will be discussed in Design Considerations of Data Augmentation.

Data Augmentations based on Deep Learning

Feature space augmentation

All of the augmentation methods discussed above are applied to images in the input space. Neural networks are incredibly powerful at mapping high-dimensional inputs into lower-dimensional representations. These networks can map images to binary classes or to $n \times 1$ vectors in flattened layers. The sequential processing of neural networks can be manipulated such that the intermediate representations can be separated from the network as a whole. The lower-dimensional representations of image data in fully-connected layers can be extracted and isolated. Konno and Iwazume [74] find a performance boost on CIFAR-100 from 66 to 73% accuracy by manipulating the modularity of neural networks to isolate and refine individual layers after training. Lower-dimensional representations found in high-level layers of a CNN are known as the feature space. DeVries and Taylor [75] presented an interesting paper discussing augmentation in this feature space. This opens up opportunities for many vector operations for Data Augmentation.

SMOTE is a popular augmentation used to alleviate problems with class imbalance. This technique is applied to the feature space by joining the k nearest neighbors to form new instances. DeVries and Taylor discuss adding noise, interpolating, and extrapolating as common forms of feature space augmentation (Figs. 13, 14).

Table 3 Performance results of the experiment with feature vs. input space extrapolation on MNIST and CIFAR-10 [75]

Model	MNIST	CIFAR-10
Baseline	1.093 ± 0.057	30.65 ± 0.27
Baseline + input space affine transformations	1.477 ± 0.068	–
Baseline + input space extrapolation	1.010 ± 0.065	–
Baseline + feature space extrapolation	*0.950 ± 0.036*	*29.24 ± 0.27*

The italic value denote high performance according to the comparative metrics

The use of auto-encoders is especially useful for performing feature space augmentations on data. Autoencoders work by having one half of the network, the encoder, map images into low-dimensional vector representations such that the other half of the network, the decoder, can reconstruct these vectors back into the original image. This encoded representation is used for feature space augmentation.

DeVries and Taylor [75] tested their feature space augmentation technique by extrapolating between the 3 nearest neighbors per sample to generate new data and compared their results against extrapolating in the input space and using affine transformations in the input space (Table 3).

Feature space augmentations can be implemented with auto-encoders if it is necessary to reconstruct the new instances back into input space. It is also possible to do feature space augmentation solely by isolating vector representations from a CNN. This is done by cutting off the output layer of the network, such that the output is a low-dimensional vector rather than a class label. Vector representations are then found by training a CNN and then passing the training set through the truncated CNN. These vector representations can be used to train any machine learning model from Naive Bayes, Support Vector Machine, or back to a fully-connected multilayer network. The effectiveness of this technique is a subject for future work.

A disadvantage of feature space augmentation is that it is very difficult to interpret the vector data. It is possible to recover the new vectors into images using an auto-encoder network; however, this requires copying the entire encoding part of the CNN being trained. For deep CNNs, this results in massive auto-encoders which are very difficult and time-consuming to train. Finally, Wong et al. [76] find that when it is possible to transform images in the data-space, data-space augmentation will outperform feature space augmentation.

Adversarial training

One of the solutions to search the space of possible augmentations is adversarial training. Adversarial training is a framework for using two or more networks with contrasting objectives encoded in their loss functions. This section will discuss using adversarial training as a search algorithm as well as the phenomenon of adversarial attacking. Adversarial attacking consists of a rival network that learns augmentations to images that result in misclassifications in its rival classification network. These adversarial attacks, constrained to noise injections, have been surprisingly successful from the perspective of the adversarial network. This is surprising because it completely defies intuition about how these models represent images. The adversarial

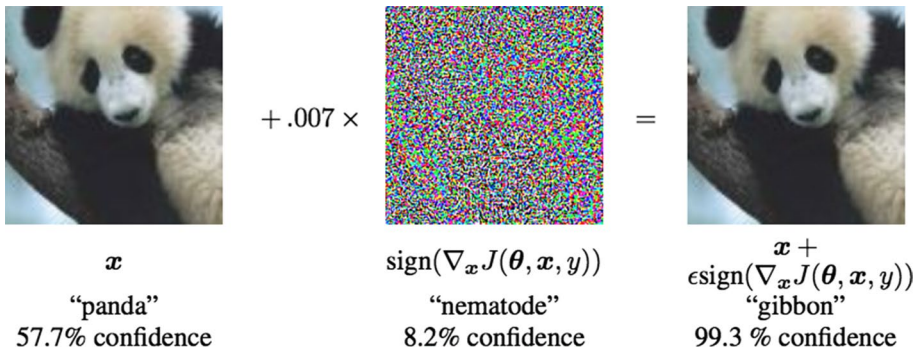

Fig. 15 Adversarial misclassification example [81]

attacks demonstrate that representations of images are much less robust than what might have been expected. This is well demonstrated by Moosavi-Dezfooli et al. [77] using DeepFool, a network that finds the minimum possible noise injection needed to cause a misclassification with high confidence. Su et al. [78] show that 70.97% of images can be misclassified by changing just one pixel. Zajac et al. [79] cause misclassifications with adversarial attacks limited to the border of images. The success of adversarial attacks is especially exaggerated as the resolution of images increases.

Adversarial attacking can be targeted or untargeted, referring to the deliberation in which the adversarial network is trying to cause misclassifications. Adversarial attacks can help to illustrate weak decision boundaries better than standard classification metrics can.

In addition to serving as an evaluation metric, defense to adversarial attacks, adversarial training can be an effective method for searching for augmentations.

By constraining the set of augmentations and distortions available to an adversarial network, it can learn to produce augmentations that result in misclassifications, thus forming an effective search algorithm. These augmentations are valuable for strengthening weak spots in the classification model. Therefore, adversarial training can be an effective search technique for Data Augmentation. This is in heavy contrast to the traditional augmentation techniques described previously. Adversarial augmentations may not represent examples likely to occur in the test set, but they can improve weak spots in the learned decision boundary.

Engstrom et al. [80] showed that simple transformations such as rotations and translations can easily cause misclassifications by deep CNN models. The worst out of the random transformations reduced the accuracy of MNIST by 26%, CIFAR10 by 72% and ImageNet (Top 1) by 28%. Goodfellow et al. [81] generate adversarial examples to improve performance on the MNIST classification task. Using a technique for generating adversarial examples known as the "fast gradient sign method", a maxout network [82] misclassified 89.4% of adversarial examples with an average confidence of 97.6%. This test is done on the MNIST dataset. With adversarial training, the error rate of adversarial examples fell from 89.4% to 17.9% (Fig. 15).

Li et al. [83] experiment with a novel adversarial training approach and compare the performance on original testing data and adversarial examples. The results displayed

Table 4 Test accuracies showing the impact of adversarial training, clean refers to the original testing data, FGSM refers to adversary examples derived from Fast Gradient Sign Method and PGD refers to adversarial examples derived from Projected Gradient Descent [83]

Models	MNIST			CIFAR-10		
	Clean	FGSM	PGD	Clean	FGSM	PGD
Standard	0.9939	0.0922	0	0.9306	0.5524	0.0256
Adversarially trained	0.9932	0.9492	0.0612	0.8755	0.8526	0.1043
Our method	0.9903	0.9713	0.9171	0.8714	0.6514	0.3440

below show how anticipation of adversarial attacks in the training process can dramatically reduce the success of attacks.

As shown in Table 4, the adversarial training in their experiment did not improve the test accuracy. However, it does significantly improve the test accuracy of adversarial examples. Adversarial defense is a very interesting subject for evaluating security and robustness of Deep Learning models. Improving on the Fast Gradient Sign Method, DeepFool, developed by Moosavi-Dezfooli et al. [77], uses a neural network to find the smallest possible noise perturbation that causes misclassifications.

Another interesting framework that could be used in an adversarial training context is to have an adversary change the labels of training data. Xie et al. [84] presented DisturbLabel, a regularization technique that randomly replaces labels at each iteration. This is a rare example of adding noise to the loss layer, whereas most of the other augmentation methods discussed add noise into the input or hidden representation layers. On the MNIST dataset with LeNet [28] CNN architecture, DisturbLabel produced a 0.32% error rate compared to a baseline error rate of 0.39%. DisturbLabel combined with Dropout Regularization produced a 0.28% error rate compared to the 0.39% baseline. To translate this to the context of adversarial training, one network takes in the classifier's training data as input and learns which labels to flip to maximize the error rate of the classification network.

The effectiveness of adversarial training in the form of noise or augmentation search is still a relatively new concept that has not been widely tested and understood. Adversarial search to add noise has been shown to improve performance on adversarial examples, but it is unclear if this is useful for the objective of reducing overfitting. Future work seeks to expand on the relationship between resistance to adversarial attacks and actual performance on test datasets.

GAN-based Data Augmentation

Another exciting strategy for Data Augmentation is generative modeling. Generative modeling refers to the practice of creating artificial instances from a dataset such that they retain similar characteristics to the original set. The principles of adversarial training discussed above have led to the very interesting and massively popular generative modeling framework known as GANs. Bowles et al. [85] describe GANs as a way to "unlock" additional information from a dataset. GANs are not the only generative

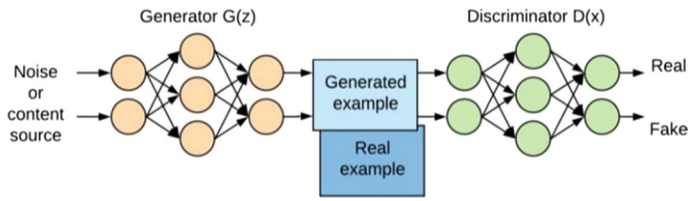

Fig. 16 Illustration of GAN concept provided by Mikolajczyk and Grochowski [72]

modeling technique that exists; however they are dramatically leading the way in computation speed and quality of results.

Another useful strategy for generative modeling worth mentioning is variational auto-encoders. The GAN framework can be extended to improve the quality of samples produced with variational auto-encoders [86]. Variational auto-encoders learn a low-dimensional representation of data points. In the image domain, this translates an image tensor of size $height \times width \times color$ channels down into a vector of size $n \times 1$, identical to what was discussed with respect to feature space augmentation. Low-dimensional constraints in vector representations will result in a poorer representation, although these constraints are better for visualization using methods such as t-SNE [87]. Imagine a vector representation of size 5×1 created by an autoencoder. These autoencoders can take in a distribution of labeled data and map them into this space. These classes could include 'head turned left', 'centered head', and 'head turned right'. The auto-encoder learns a low-dimensional representation of these data points such that vector operations such as adding and subtracting can be used to simulate a front view-3D rotation of a new instance. Variational auto-encoder outputs can be further improved by inputting them into GANs [31]. Additionally, a similar vector manipulation process can be done on the noise vector inputs to GANs through the use of Bidirectional GANs [88].

The impressive performance of GANs has resulted in increased attention on how they can be applied to the task of Data Augmentation. These networks have the ability to generate new training data that results in better performing classification models. The GAN architecture first proposed by Ian Goodfellow [31] is a framework for generative modeling through adversarial training. The best anecdote for understanding GANs is the analogy of a cop and a counterfeiter. The counterfeiter (generator network) takes in some form of input. This could be a random vector, another image, text, and many more. The counterfeiter learns to produce money such that the cop (discriminator network) cannot tell if the money is real or fake. The real or fake dichotomy is analogous to whether or not the generated instance is from the training set or if it was created by the generator network (Fig. 16).

The counterfeiter versus robber analogy is a seamless bridge to understand GANs in the context of network intrusion detection. Lin et al. [89] use a generator network to learn how to fool a black-box detection system. This highlights one of the most interesting characteristics of GANs. Analysis tools derived from game theory such as minimax strategy and the Nash Equilibrium [90] suggest that the generator will eventually fool the discriminator. The success of the generator to overcome the discriminator makes it very powerful for generative modeling. GANs are the most promising generative modeling technique for use in Data Augmentation.

The vanilla GAN architecture uses multilayer perceptron networks in the generator and discriminator networks. This is able to produce acceptable images on a simple image dataset such as the MNIST handwritten digits. However, it fails to produce quality results for higher resolution, more complicated datasets. In the MNIST dataset, each image is only $28 \times 28 \times 1$ for a total of 784 pixels. GANs applied to the MNIST data are able to produce convincing results. However, MNIST images are far less challenging than other image datasets due to low intra-class variance and resolution, to name a couple differences of many. This is in heavy contrast with other datasets studied in most academic Computer Vision papers such as ImageNet or CIFAR-10. For immediate reference, an ImageNet image is of resolution $256 \times 256 \times 3$, totaling 196,608 pixels, a $250\times$ increase in pixel count compared with MNIST.

Many research papers have been published that modify the GAN framework through different network architectures, loss functions, evolutionary methods, and many more. This research has significantly improved the quality of samples created by GANs. There have been many new architectures proposed for expanding on the concept of GANs and producing higher resolution output images, many of which are out of the scope of this paper. Amongst these new architectures, DCGANs, Progressively Growing GANs, CycleGANs, and Conditional GANs seem to have the most application potential in Data Augmentation.

The DCGAN [91] architecture was proposed to expand on the internal complexity of the generator and discriminator networks. This architecture uses CNNs for the generator and discriminator networks rather than multilayer perceptrons. The DCGAN was tested to generate results on the LSUN interior bedroom image dataset, each image being $64 \times 64 \times 3$, for a total of 12,288 pixels, (compared to 784 in MNIST). The idea behind DCGAN is to increase the complexity of the generator network to project the input into a high dimensional tensor and then add deconvolutional layers to go from the projected tensor to an output image. These deconvolutional layers will expand on the spatial dimensions, for example, going from $14 \times 14 \times 6$ to $28 \times 28 \times 1$, whereas a convolutional layer will decrease the spatial dimensions such as going from $14 \times 14 \times 32$ to $7 \times 7 \times 64$. The DCGAN architecture presents a strategy for using convolutional layers in the GAN framework to produce higher resolution images (Figs. 17, 18).

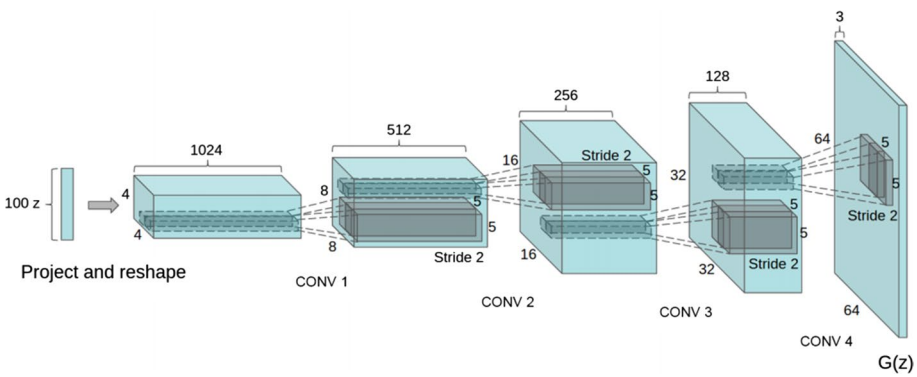

Fig. 17 DCGAN, generator architecture presented by Radford et al. [91]

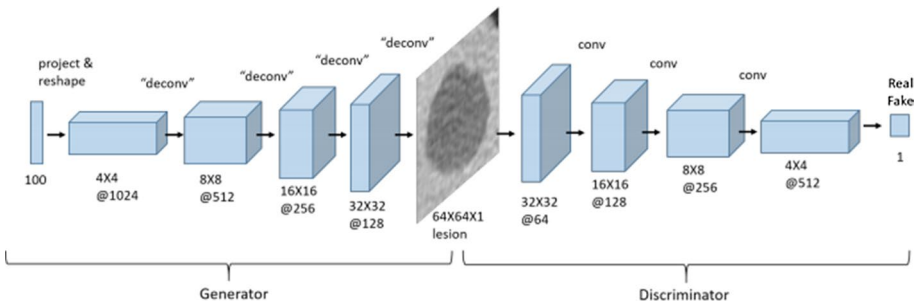

Fig. 18 Complete DCGAN architecture used by Frid-Adar et al. [49] to generate liver lesion images

Frid-Adar et al. [49] tested the effectiveness of using DCGANs to generate liver lesion medical images. They use the architecture pictured above to generate $64 \times 64 \times 1$ size images of liver lesion CT scans. Their original dataset contains 182 CT scans, (53 Cysts, 64 Metastases, and 65 Hemangiomas). After using classical augmentations to achieve 78.6% sensitivity and 88.4% specificity, they observed an increase to 85.7% sensitivity and 92.4% specificity once they added the DCGAN-generated samples.

Another architecture of interest is known as Progressively Growing GANs [34]. This architecture trains a series of networks with progressive resolution complexity. These resolutions range from 4×4 to 8×8 and so on until outputs of size 1024×1024 are achieved. This is built on the concept that GANs can accept images as input as well as random vectors. Therefore, the series of GANs work by passing samples from a lower resolution GAN up to higher-resolution GANs. This has produced very amazing results on facial images.

In addition to improving the resolution size of GANs, another interesting architecture that increases the quality of outputs is the CycleGAN [92] proposed by Zhu et al. CycleGAN introduces an additional Cycle-Consistency loss function to help stabilize GAN training. This is applied to image-to-image translation. Neural Style Transfer [32], discussed further in the section below, learns a single image to single image translation. However, CycleGAN learns to translate from a domain of images to another domain, such as horses to zebras. This is implemented via forward and backward consistency loss functions. A generator takes in images of horses and learns to map them to zebras such that the discriminator cannot tell if they were originally a part of the zebra set or not, as discussed above. After this, the generated zebras from horse images are passed through a network which translates them back into horses. A second discriminator determines if this re-translated image belongs to the horse set or not. Both of these discriminator losses are aggregated to form the cycle-consistency loss.

The use of CycleGANs was tested by Zhu et al. [93] in the task of Emotion Classification. Using the emotion recognition dataset, FER2013 [94], Facial Expression Recognition Database, they build a CNN classifier to recognize 7 different emotions: angry, disgust, fear, happy, sad, surprise, and neutral. These classes are imbalanced and the CycleGAN is used as a method of intelligent oversampling.

CycleGANs learned an unpaired image-to-image translation between domains. An example of the domains in this problem is neutral to disgust. The CycleGAN learns to translate an image representing a neutral image into an image representing the disgust emotion (Figs. 19, 20).

Fig. 19 Some examples of synthetic data created with CycleGANs for emotion classification

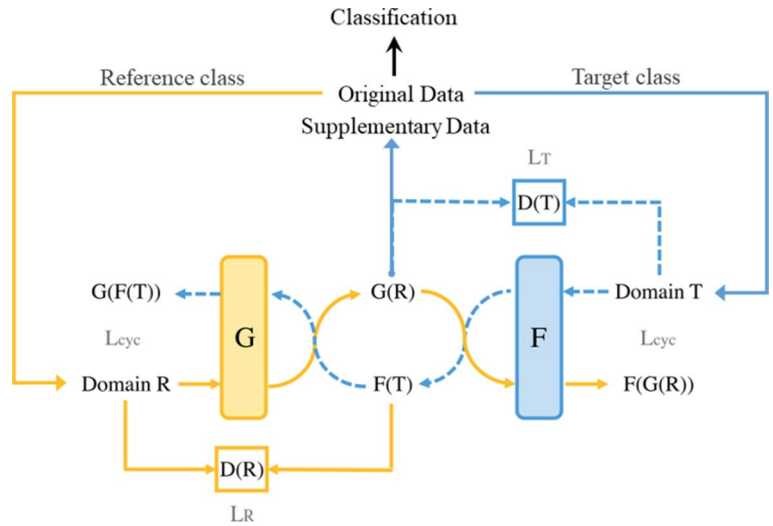

Fig. 20 Architecture overview: G and F consist of two separate GANs composing the CycleGAN. Two images are taken from the reference and target class and used to generate new data in the target class

Using CycleGANs to translate images from the other 7 classes into the minority classes was very effective in improving the performance of the CNN model on emotion recognition. Employing these techniquess, accuracy improved 5–10%. To further understand the effectiveness of adding GAN-generated instances, a t-SNE visualization is used. t-SNE [87] is a visualization technique that learns to map between high-dimensional vectors into a low-dimensional space to facilitate the visualization of decision boundaries (Fig. 21).

Another interesting GAN architecture for use in Data Augmentation is Conditional GANs [95]. Conditional GANs add a conditional vector to both the generator and the discriminator in order to alleviate problems with mode collapse. In addition to inputting a random vector z to the generator, Conditional GANs also input a y vector which could be something like a one-hot encoded class label, e.g. [0 0 0 1 0]. This class label targets a specific class for the generator and the discriminator (Fig. 22).

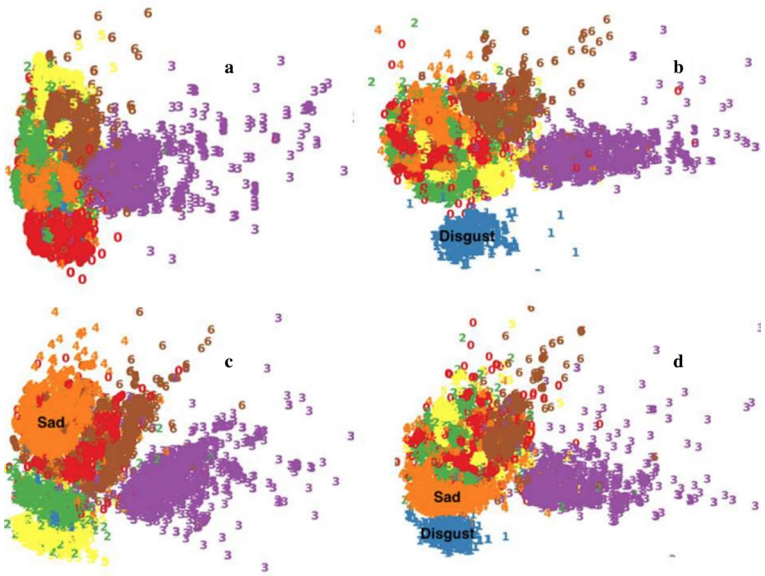

Fig. 21 t-SNE visualization demonstrating the improved decision boundaries when using CycleGAN-generated samples. **a** original CNN model, **b** adding GAN-generated disgust images, **c** adding GAN-generated sad images, **d** adding both GAN-generated disgust and sad images [93]

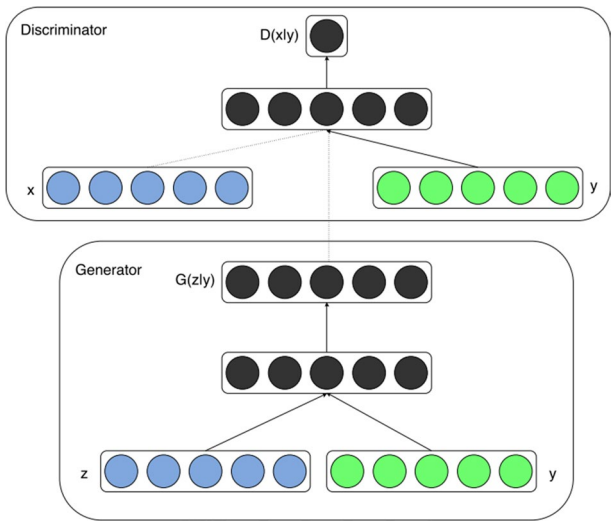

Fig. 22 Illustration from Mirza and Osindero [95] showing how the conditional y vector is integrated into the GAN framework

Lucic et al. [96] sought out to compare newly developed GAN loss functions. They conducted a series of tests that determined most loss functions can reach similar scores with enough hyperparameter optimization and random restarts. This suggests that increased computational power is a more promising area of focus than algorithmic changes in the generator versus discriminator loss function.

Most of the research done in applying GANs to Data Augmentation and reporting the resulting classification performance has been done in biomedical image analysis

[39]. These papers have shown improved classification boundaries derived from training with real and generated data from GAN models. In addition, some papers measure the quality of GAN outputs by a visual Turing test. In these tests, the study asks two experts to distinguish between real and artificial images in medical image tasks such as skin lesion classification and liver cancer detection. Table 5 shows that the first and second experts were only able to correctly label 62.5% and 58.6% of the GAN-generated liver lesion images as fake. Labeling images as fake refers to their origin coming from the generator rather than an actual liver lesion image (Table 6; Fig. 23).

Table 5 Results of 'Visual Turing Test' on DCGAN-generated liver lesion images presented by Frid-Adar et al. [139]

	Classification accuracy			Is ROI real?
	Real (%)	Synthetic (%)	Total score	Total score
Expert 1	78	77.5	235\302 = 77.8%	189\302 = 62.5%
Expert 2	69.2	69.2	209\302 = 69.2%	177\302 = 58.6%

Table 6 Results of 'Visual Turing Test' on different DCGAN- and WGAN [104]—generated brain tumor MR images presented by Han et al. [140]

	Accuracy (%)	Real selected as real	Real as synt	Synt as real	Synt as synt
T1 (DCGAN, 128 × 128)	70	26	24	6	44
Tlc (DCGAN, 128 × 128)	71	24	26	3	47
T2 (DCGAN, 128 × 128)	64	22	28	8	42
FLAIR (DCGAN, 128 × 128)	54	12	38	8	42
Concat (DCGAN, 128 × 128)	77	34	16	7	43
Concat (DCGAN, 64 × 64)	54	13	37	9	41
T1 (WGAN, 128 × 128)	64	20	30	6	44
Tlc (WGAN, 128 × 128)	55	13	37	8	42
T2 (WGAN, 128 × 128)	58	19	31	11	39
FLAIR (WGAN, 128 × 128)	62	16	34	4	46
Concat (WGAN, 128 × 128)	66	31	19	15	35
Concat (WGAN, 64 × 64)	53	18	32	15	35

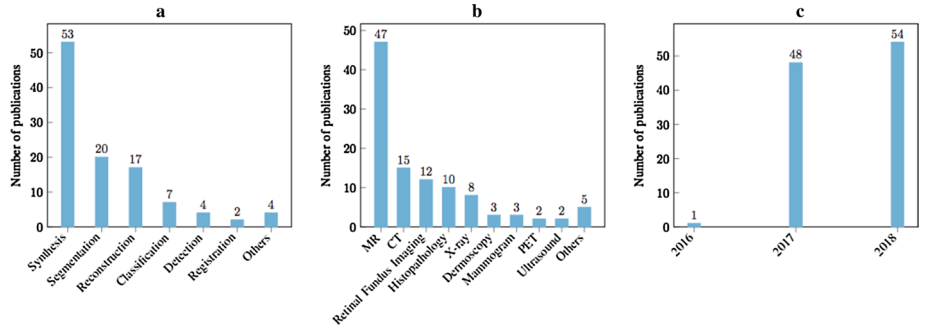

Fig. 23 Trends in applying GANs to medical image analysis [39]

Fig. 24 Adversarial autoencoder framework used in DOPING [97]

GAN samples can be used as an oversampling technique to solve problems with class imbalance. Lim et al. [97] show how GAN samples can be used for unsupervised anomaly detection. By oversampling rare normal samples, which are samples that occur with small probability, GANs are able to reduce the false positive rate of anomaly detection. They do this using the Adversarial Autoencoder framework proposed by Makhzani et al. [98] (Fig. 24).

As exciting as the potential of GANs is, it is very difficult to get high-resolution outputs from the current cutting-edge architectures. Increasing the output size of the images produced by the generator will likely cause training instability and non-convergence. Another drawback of GANs is that they require a substantial amount of data to train. Thus, depending on how limited the initial dataset is, GANs may not be a practical solution. Salimans et al. [99] provide a more complete description of the problems with training GANs.

Neural Style Transfer

Neural Style Transfer [32] is one of the flashiest demonstrations of Deep Learning capabilities. The general idea is to manipulate the representations of images created in CNNs. Neural Style Transfer is probably best known for its artistic applications, but it also serves as a great tool for Data Augmentation. The algorithm works by manipulating the sequential representations across a CNN such that the style of one image can be transferred to another while preserving its original content. A more detailed explanation of the gram matrix operation powering Neural Style Transfer can be found by Li et al. [100] (Fig. 25).

It is important to also recognize an advancement of the original algorithm from Gatys et al. known as Fast Style Transfer [35]. This algorithm extends the loss function from a per-pixel loss to a perceptual loss and uses a feed-forward network to stylize images. This perceptual loss is reasoned about through the use of another pre-trained net. The use of perceptual loss over per-pixel loss has also shown great promise in the application of super-resolution [101] as well as style transfer. This loss function enhancement enables style transfer to run much faster, increasing interest in practical applications. Additionally, Ulyanov et al. [102] find that replacing batch normalization with instance normalization results in a significant improvement for fast stylization (Fig. 26).

For the purpose of Data Augmentation, this is somewhat analogous to color space lighting transformations. Neural Style Transfer extends lighting variations and enables the encoding of different texture and artistic styles as well. This leaves practitioners of Data Augmentation with the decision of which styles to sample from when deriving new images via Neural Style Transfer.

Choosing which styles to sample from can be a challenging task. For applications such as self-driving cars it is fairly intuitive to think of transferring training

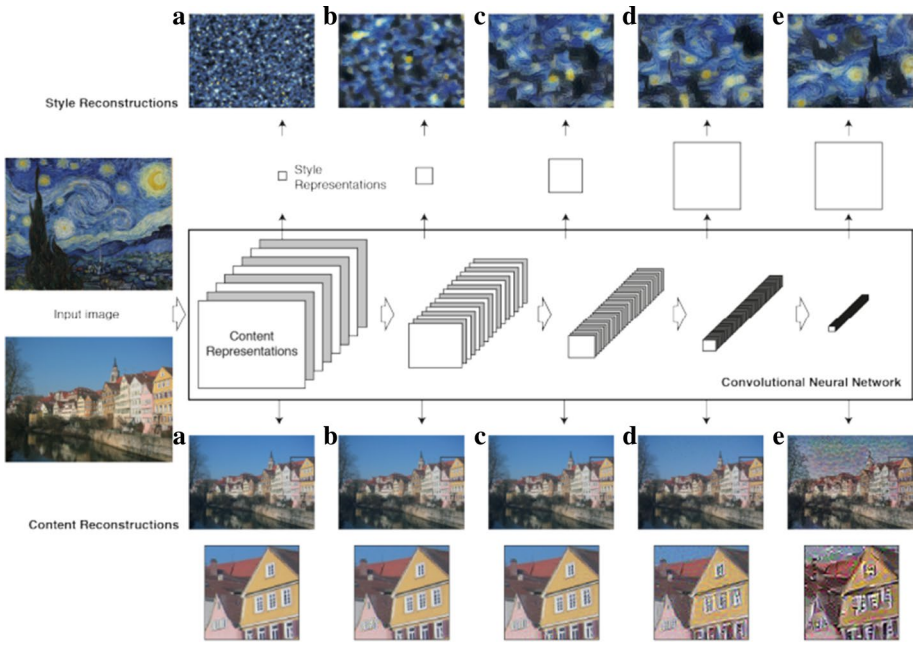

Fig. 25 Illustration of style and content reconstructions in Neural Style Transfer [32]

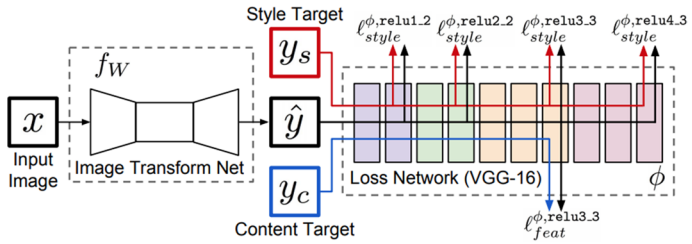

Fig. 26 Illustration of the Fast neural style algorithm by Johnson et al. [35]

data into a night-to-day scale, winter-to-summer, or rainy-to-sunny scale. However, in other application domains, the set of styles to transfer into is not so obvious. For ease of implementation, data augmentation via Neural Style Transfer could be done by selecting a set of k styles and applying them to all images in the training set. The work of Style Augmentation [103], avoids introducing a new form of style bias into the dataset by deriving styles at random from a distribution of 79,433 artistic images. Transferring style in training data has been tested on the transition from simulated environments to the real-world. This is very useful for robotic manipulation tasks using Reinforcement Learning because of potential damages to hardware when training in the real-world. Many constraints such as low-fidelity cameras cause these models to generalize poorly when trained in physics simulations and deployed in the real-world.

Tobin et al. [104] explore the effectiveness of using different styles in training simulation and achieve within 1.5 cm accuracy in the real-world on the task of object localization. Their experiments randomize the position and texture of the objects to be detected

Fig. 27 Examples of different styles simulated by Tobin et al. [104]

on the table in the simulation, as well as the texture, lighting, number of lights, and random noise in the background. They found that with enough variability in the training data style, the real-world simply appears as another variation to the model. Interestingly, they found that diversity in styles was more effective than simulating in as realistic of an environment as possible. This is in contrast to the work of Shrivastava et al. [105] who used GANs to make their simulated data as realistic as possible (Fig. 27).

Using simulated data to build Computer Vision models has been heavily investigated. One example of this is from Richter et al. [106]. They use computer graphics from modern open-world games such as Grand Theft Auto to produce semantic segmentation datasets. The authors highlight anecdotes of the manual annotation costs required to build these pixel-level datasets. They mention the CamVid dataset [107] requires 60 min per image to manually annotate, and the Cityscapes dataset [108] requires 90 min per image. This high labor and time cost motivates the use and development of synthetic datasets. Neural Style Transfer is a very interesting strategy to improve the generalization ability of simulated datasets.

A disadvantage of Neural Style Transfer Data Augmentation is the effort required to select styles to transfer images into. If the style set is too small, further biases could be introduced into the dataset. Trying to replicate the experiments of Tobin et al. [104] will require a massive amount of additional memory and compute to transform and store 79,433 new images from each image. The original algorithm proposed by Gatys et al. [32] has a very slow running time and is therefore not practical for Data Augmentation. The algorithm developed by Johnson et al. [35] is much faster, but limits transfer to a pre-trained set of styles.

Meta learning Data Augmentations

The concept of meta-learning in Deep Learning research generally refers to the concept of optimizing neural networks with neural networks. This approach has become very popular since the publication of NAS [33] from Zoph and Le. Real et al. [109, 110] also

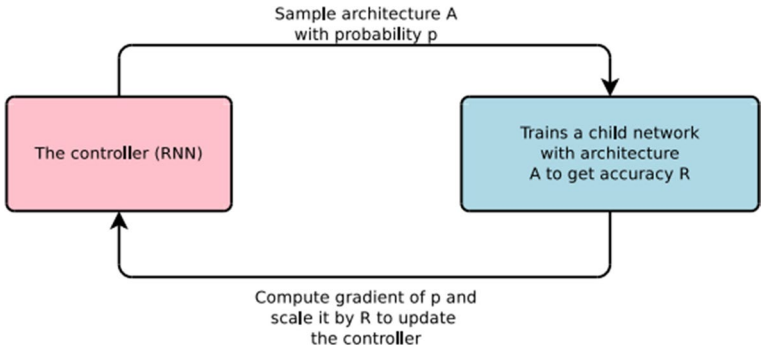

Fig. 28 Concept behind Neural Architecture Search [33]

show the effectiveness of evolutionary algorithms for architecture search. Salimans et al. [111] directly compare evolutionary strategies with Reinforcement Learning. Another interesting alternative to Reinforcement Learning is simple random search [112]. Utilizing evolutionary and random search algorithms is an interesting area of future work, but the meta-learning schemes reviewed in this survey are all neural-network, gradient-based.

The history of Deep Learning advancement from feature engineering such as SIFT [113] and HOG [114] to architecture design such as AlexNet [1], VGGNet [2], and Inception-V3 [4], suggest that meta-architecture design is the next paradigm shift. NAS takes a novel approach to meta-learning architectures by using a recurrent network trained with Reinforcement Learning to design architectures that result in the best accuracy. On the CIFAR-10 dataset, this achieved an error rate of 3.65 (Fig. 28).

This section will introduce three experiments using meta-learning for Data Augmentation. These methods use a prepended neural network to learn Data Augmentations via mixing images, Neural Style Transfer, and geometric transformations.

Neural augmentation The Neural Style Transfer algorithm requires two parameters for the weights of the style and content loss. Perez and Wang [36] presented an algorithm to meta-learn a Neural Style Transfer strategy called Neural Augmentation. The Neural Augmentation approach takes in two random images from the same class. The prepended augmentation net maps them into a new image through a CNN with 5 layers, each with 16 channels, 3×3 filters, and ReLU activation functions. The image outputted from the augmentation is then transformed with another random image via Neural Style Transfer. This style transfer is carried out via the CycleGAN [92] extension of the GAN [31] framework. These images are then fed into a classification model and the error from the classification model is backpropagated to update the Neural Augmentation net. The Neural Augmentation network uses this error to learn the optimal weighting for content and style images between different images as well as the mapping between images in the CNN (Fig. 29).

Perez and Wang tested their algorithm on the MNIST and Tiny-imagenet-200 datasets on binary classification tasks such as cat versus dog. The Tiny-imagenet-200 dataset is used to simulate limited data. The Tiny-imagenet-200 dataset contains only 500 images in each of the classes, with 100 set aside for validation. This problem

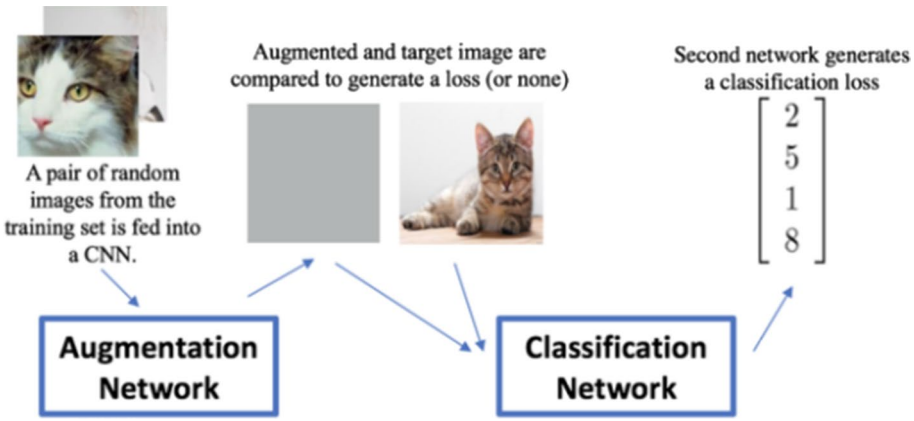

Fig. 29 Illustration of augmentation network [36]

limits this dataset to 2 classes. Thus there are only 800 images for training. Each of the Tiny-imagenet-200 images is $64 \times 64 \times 3$, and the MNIST images are $28 \times 28 \times 1$. The experiment compares their proposed Neural Augmentation [36] approach with traditional augmentation techniques such as cropping and rotation, as well as with a style transfer approach with a predetermined set of styles such as Night/Day and Winter/Summer.

The traditional baseline study transformed images by choosing an augmentation from a set (shifted, zoomed in/out, rotated, flipped, distorted, or shaded with a hue). This was repeated to increase the dataset size from N to 2 N. The GAN style transfer baseline uses 6 different styles to transform images (Cezanne, Enhance, Monet, Ukiyoe, Van Gogh and Winter). The Neural Augmentation techniques tested consist of three levels based on the design of the loss function for the augmentation net (Content loss, Style loss via gram matrix, and no loss computer at this layer). All experiments are tested with a convolutional network consisting of 3 convolutional layers each followed by max pooling and batch normalization, followed by 2 fully-connected layers. Each experiment runs for 40 epochs at a learning rate of 0.0001 with the Adam optimization technique (Table 7).

The results of the experiment are very promising. The Neural Augmentation technique performs significantly better on the Dogs versus Goldfish study and only slightly worse on Dogs versus Cats. The technique does not have any impact on the MNIST problem. The paper suggests that the likely best strategy would be to combine the traditional augmentations and the Neural Augmentations.

Smart Augmentation The Smart Augmentation [37] approach utilizes a similar concept as the Neural Augmentation technique presented above. However, the combination of images is derived exclusively from the learned parameters of a prepended CNN, rather than using the Neural Style Transfer algorithm.

Smart Augmentation is another approach to meta-learning augmentations. This is done by having two networks, *Network-A* and *Network-B*. *Network-A* is an augmentation network that takes in two or more input images and maps them into a new image or images to train *Network-B*. The change in the error rate in *Network-B* is then

Table 7 Results comparing augmentations [36]

Quantitative results on dogs vs. goldfish	
Dogs vs goldfish	
Augmentation	Val. acc.
None	0.855
Traditional	0.890
GANs	0.865
Neural + no loss	*0.915*
Neural + content loss	0.900
Neural + style	0.890
Control	0.840
Quantitative results on dogs vs cats	
Dogs vs cat	
Augmentation	Val. acc.
None	0.705
Traditional	*0.775*
GANs	0.720
Neural + no loss	0.765
Neural + content loss	0.770
Neural + style	0.740
Control	0.710
MNIST 0's and 8's	
Augmentation	Val. acc.
None	0.972
Neural + no loss	*0.975*
Neural + content loss	0.968

backpropagated to update *Network-A*. Additionally another loss function is incorporated into *Network-A* to ensure that its outputs are similar to others within the class. *Network-A* uses a series of convolutional layers to produce the augmented image. The conceptual framework of *Network-A* can be expanded to use several Networks trained in parallel. Multiple *Network-As* could be very useful for learning class-specific augmentations via meta-learning (Fig. 30).

Smart Augmentation is similar to SamplePairing [65] or mixed-examples in the sense that a combination of existing examples produces new ones. However, the mechanism of Smart Augmentation is much more sophisticated, using an adaptive CNN to derive new images rather than averaging pixels or hand-engineered image combinations.

The Smart Augmentation technique was tested on the task of gender recognition. On the Feret dataset, accuracy improved from 83.52 to 88.46%. The audience dataset responded with an improvement of 70.02% to 76.06%. Most interestingly, results from another face dataset increased from 88.15 to 95.66%. This was compared with traditional augmentation techniques which increased the accuracy from 88.15 to 89.08%. Additionally, this experiment derived the same accuracy when using two *Network-As* in the augmentation framework as was found with one *Network-A*. This experiment demonstrates

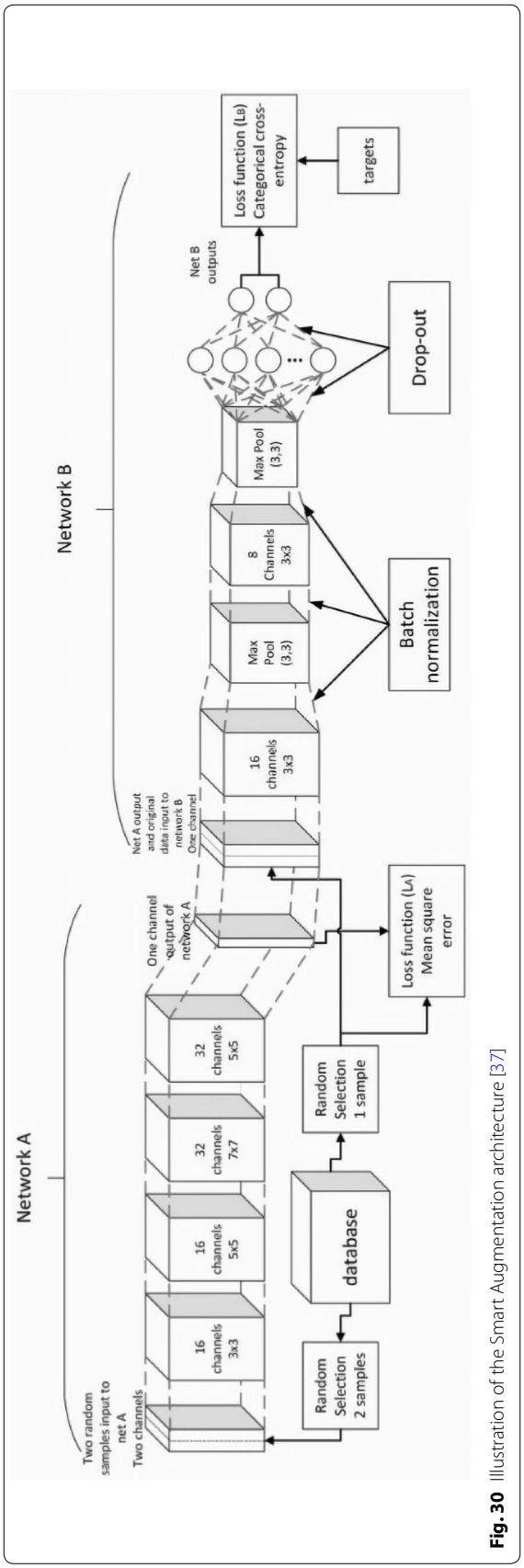

Fig. 30 Illustration of the Smart Augmentation architecture [37]

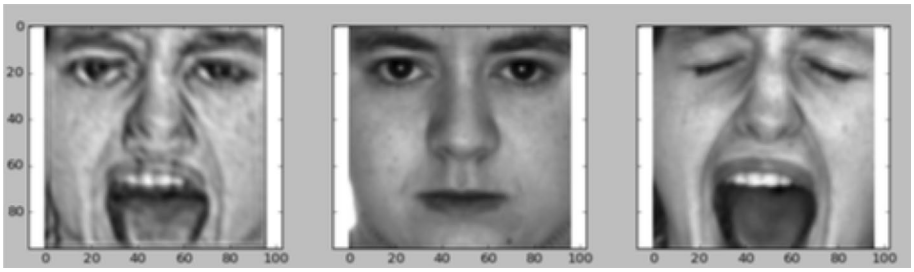

Fig. 31 On the gender recognition task, the image to the left is an example of an instance produced by Network-A in Smart Augmentation given the right images as input [37]

the significant performance increase with the Smart Augmentation meta-learning strategy (Fig. 31).

AutoAugment AutoAugment [38], developed by Cubuk et al., is a much different approach to meta-learning than Neural Augmentation or Smart Augmentation. AutoAugment is a Reinforcement Learning algorithm [115] that searches for an optimal augmentation policy amongst a constrained set of geometric transformations with miscellaneous levels of distortions. For example, 'translateX 20 pixels' could be one of the transformations in the search space (Table 8).

In Reinforcement Learning algorithms, a policy is analogous to the strategy of the learning algorithm. This policy determines what actions to take at given states to achieve some goal. The AutoAugment approach learns a policy which consists of many sub-policies, each sub-policy consisting of an image transformation and a magnitude of transformation. Reinforcement Learning is thus used as a discrete search algorithm of augmentations. The authors also suggest that evolutionary algorithms or random search would be effective search algorithms as well.

AutoAugment found policies which achieved a 1.48% error rate on CIFAR-10. AutoAugment also achieved an 83.54% Top-1 accuracy on the ImageNet dataset. Very interestingly as well, the policies learned on the ImageNet dataset were successful when transferred to the Stanford Cars and FGVC Aircraft image recognition tasks. In this case, the ImageNet policy applied to these other datasets reduced error rates by 1.16% and 1.76% respectively.

Geng et al. [116] expanded on AutoAugment by replacing the Reinforcement Learning search algorithm with Augmented Random Search (ARS) [112]. The authors point out that the sub-policies learned from AutoAugment are inherently flawed because of the discrete search space. They convert the probability and magnitude of augmentations into a continuous space and search for sub-policies with ARS. With this, they achieve lower error rates on CIFAR-10, CIFAR-100, and ImageNet (Table 9).

Minh et al. [117] also experimented with using Reinforcement Learning [115] to search for Data Augmentations. They further explore the effectiveness of learning transformations for individual instances rather than the entire dataset. They find classification accuracy differences of 70.18% versus 74.42% on the CIFAR-10 dataset and 74.61% versus 80.35% on the problem of classifying dogs versus cats. Further, they explore the robustness of classifiers with respect to test-time augmentation and find that the model

Table 8 AutoAugment augmentation policy found on the reduced CIFAR-10 dataset [38]

	Operation 1	Operation 2
Sub-policy 0	(Invert,0.1,7)	(Contrast,0.2,6)
Sub-policy 1	(Rotate,0.7,2)	(TranslateX,0.3,9)
Sub-policy 2	(Sharpness,0.8,1)	(Sharpness,0.9,3)
Sub-policy 3	(ShearY,0.5,8)	(TranslateY,0.7,9)
Sub-policy 4	(AutoContrast,0.5,8)	(Equalize,0.9,2)
Sub-policy 5	(ShearY,0.2,7)	(Posterize,0.3,7)
Sub-policy 6	(Color,0.4,3)	(Brightness,0.6,7)
Sub-policy 7	(Sharpness,0.3,9)	(Brightness,0.7,9)
Sub-policy 8	(Equalize,0.6,5)	(Equalize,0.5,1)
Sub-policy 9	(Contrast,0.6,7)	(Sharpness,0.6,5)
Sub-policy 10	(Color,0.7,7)	(TranslateX,0.5,8)
Sub-policy 11	(Equalize,0.3,7)	(AutoContrast,0.4,8)
Sub-policy 12	(TranslateY,0.4,3)	(Sharpness,0.2,6)
Sub-policy 13	(Brightness,0.9,6)	(Color,0.2,8)
Sub-policy 14	(Solarize,0.5,2)	(Invert,0.0,3)
Sub-policy 15	(Equalize,0.2,0)	(AutoContrast,0.6,0)
Sub-policy 16	(Equalize,0.2,8)	(Equalize,0.6,4)
Sub-policy 17	(Color,0.9,9)	(Equalize,0.6,6)
Sub-policy 18	(AutoContrast,0.8,4)	(Solarize,0.2,8)
Sub-policy 19	(Brightness,0.1,3)	(Color,0.7,0)
Sub-policy 20	(Solarize,0.4,5)	(AutoContrast,0.9,3)
Sub-policy 21	(TranslateY,0.9,9)	(TranslateY,0.7,9)
Sub-policy 22	(AutoContrast,0.9,2)	(Solarize,0.8,3)
Sub-policy 23	(Equalize,0.8,8)	(Invert,0.1,3)
Sub-policy 24	(TranslateY,0.7,9)	(AutoContrast,0.9,1)

Table 9 The performance of ARS on continuous space vs. AutoAugment on discrete space [116]

Model	AutoAugment	ARS-Aug
Wide-ResNet-28-10	2.68	2.33
Shake-Shake (26 2 × 32 days)	2.47	2.14
Shake-Shake (26 2 × 96 days)	1.99	1.68
Shake-Shake (26 2 × 112 days)	1.89	1.59
AmoebaNet-B (6,128)	1.75	1.49
PyramidNet + ShakeDrop	1.48	*1.26*

trained with Reinforcement Learning augmentation search performs much better. On the CIFAR-10 dataset this results in 50.99% versus 70.06% accuracy when the models are evaluated on augmented test data.

A disadvantage to meta-learning is that it is a relatively new concept and has not been heavily tested. Additionally, meta-learning schemes can be difficult and time-consuming to implement. Practitioners of meta-learning will have to solve problems primarily with vanishing gradients [118], amongst others, to train these networks.

Comparing Augmentations

As shown throughout "Design considerations for image Data Augmentation" section, possibilities for Data Augmentation. However, there are not many comparative studies that show the performance differences of these different augmentations. One such study was conducted by Shijie et al. [119] which compared GANs, WGANs, flipping, cropping, shifting, PCA jittering, color jittering, adding noise, rotation, and some combinations on the CIFAR-10 and ImageNet datasets. Additionally, the comparative study ranged across dataset sizes with the small set consisting of 2 k samples with 200 in each class, tthe medium set consisting of 10 k samples with 1 k in each class, and the large set consisting of 50 k samples with 5 k in each class. They also tested with 3 levels of augmentation, no augmentation, original plus same size of generated samples, and original plus double size of generated samples. They found that cropping, flipping, WGAN, and rotation generally performed better than others. The combinations of flipping + cropping and flipping + WGAN were the best overall, improving classification performance on CIFAR-10 by + 3% and + 3.5%, respectively.

Design considerations for image Data Augmentation

This section will briefly describe some additional design decisions with respect to Data Augmentation techniques on image data.

Test-time augmentation

In addition to augmenting training data, many research reports have shown the effectiveness of augmenting data at test-time as well. This can be seen as analogous to ensemble learning techniques in the data space. By taking a test image and augmenting it in the same way as the training images, a more robust prediction can be derived. This comes at a computational cost depending on the augmentations performed, and it can restrict the speed of the model. This could be a very costly bottleneck in models that require real-time prediction. However, test-time augmentation is a promising practice for applications such as medical image diagnosis. Radosavovic et al. [120] denote test-time augmentation as data distillation to describe the use of ensembled predictions to get a better representation of the image.

Wang et al. [121] sought out to develop a mathematical framework to formulate test-time augmentation. Testing their test-time augmentation scheme on medical image segmentation, they found that it outperformed the single-prediction baseline and dropout-based multiple predictions. They also found better uncertainty estimation when using test-time augmentation, reducing highly confident but incorrect predictions. Their test-time augmentation method uses a Monte Carlo simulation in order to obtain parameters for different augmentations such as flipping, scaling, rotation, and translations, as well as noise injections.

Test-time augmentation can be found in the Alexnet paper [1], which applies CNNs to the ImageNet dataset. In their experiments, they average the predictions on ten randomly cropped patches. These patches consist of one extracted from the center, four corner croppings, and the equivalent regions on the horizontally flipped images. These predictions are averaged to form the final output. He et al. [3] use the same 10-crop testing procedure to evaluate their ResNet CNN architecture (Fig. 32).

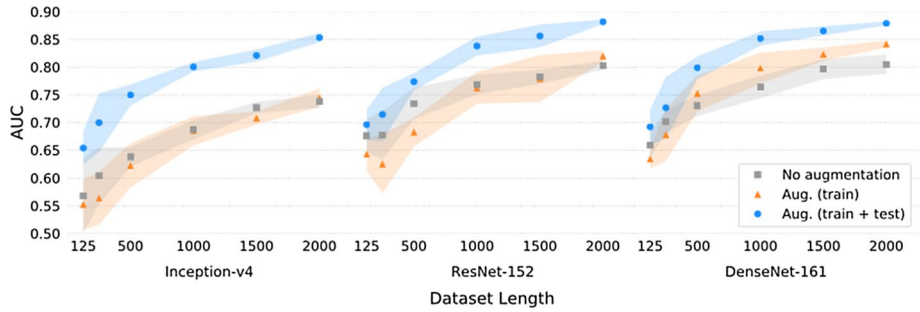

Fig. 32 Impact of test-time data augmentation for skin lesion classification [122]

Perez et al. [122] present a study on the effectiveness of test-time augmentation with many augmentation techniques. These augmentations tested include color augmentation, rotation, shearing, scaling, flipping, random cropping, random erasing, elastic, mixing, and combinations between the techniques. Table 9 shows the higher performance achieved when augmenting test images as well as training images. Matsunaga et al. [123] also demonstrate the effectiveness of test-time augmentation on skin lesion classification, using geometric transformations such as rotation, translation, scaling, and flipping.

The impact of test-time augmentation on classification accuracy is another mechanism for measuring the robustness of a classifier. A robust classifier is thus defined as having a low variance in predictions across augmentations. For example, a prediction of an image should not be much different when that same image is rotated 20°. In their experiments searching for augmentations with Reinforcement Learning, Minh et al. [117] measure robustness by distorting test images with a 50% probability and contrasting the accuracy on un-augmented data with the augmented data. In this study, the performance of the baseline model decreases from 74.61 to 66.87% when evaluated on augmented test images.

Some classification models lie on the fence in terms of their necessity for speed. This suggests promise in developing methods that incrementally upgrade the confidence of prediction. This could be done by first outputting a prediction with little or no test-time augmentation and then incrementally adding test-time augmentations to increase the confidence of the prediction. Different Computer Vision tasks require certain constraints on the test-time augmentations that can be used. For example, image recognition can easily aggregate predictions across warped images. However, it is difficult to aggregate predictions on geometrically transformed images in object detection and semantic segmentation.

Curriculum learning

Aside from the study of Data Augmentation, many researchers have been interested in trying to find a strategy for selecting training data that beats random selection. In the context of Data Augmentation, research has been published investigating the relationship between original and augmented data across training epochs. Some research

Table 10 Comparison of resolution across three very popular open-source image datasets

Dataset	Resolution
MNIST handwritten digits	$28 \times 28 \times 1$
CIFAR-10/100	$32 \times 32 \times 3$
ImageNet	$256 \times 256 \times 3$

suggests that it is best to initially train with the original data only and then finish training with the original and augmented data, although there is no clear consensus.

In the SamplePairing [65] study, one epoch on ImageNet and 100 epochs on other datasets are completed without SamplePairing before mixed image data is added to the training. Once the SamplePairing images are added to the training set, they run in cycles between 8:2 epochs, 8 with SamplePairing images, 2 without. Jaderberg et al. [124] train exclusively with synthetic data for natural scene text recognition. The synthetic data produced the training data by enumerating through different fonts and augmentations. This produced sets of training images for size 50 k and 90 k lexicons. Mikolajczyk and Grochowski [72] draw comparisons from transfer learning. They suggest that training on augmented data to learn the initial weights of a deep convolutional network is similar to transferring weights trained on other datasets such as ImageNet. These weights are then fine-tuned only with the original training data.

Curriculum learning decisions are especially important for One-Shot Learning systems such as FaceNet, presented by Schroff et al. [125]. It is important to find faces which are somewhat similar to the new face such that the learned distance function is actually useful. In this sense, the concept of curriculum learning shares many similarities with adversarial search algorithms or learning only on hard examples.

Curriculum learning, a term originally coined by Bengio et al. [126], is an applicable concept for all Deep Learning models, not just those constrained with limited data. Plotting out training accuracy over time across different initial training subsets could help reveal patterns in the data that dramatically speed up training time. Data Augmentation constructs massively inflated training from combinations such as flipping, translating, and randomly erasing. It is highly likely that a subset exists in this set such that training will be faster and more accurate.

Resolution impact

Another interesting discussion about Data Augmentation in images is the impact of resolution. Higher resolution images such as HD ($1920 \times 1080 \times 3$) or 4 K ($3840 \times 2160 \times 3$) require much more processing and memory to train deep CNNs. However, it seems intuitive that next-generation models would be trained on higher resolution images. Many current models downsample images from their original resolution to make the classification problem computationally more feasible. However, sometimes this downsampling causes information loss within the image, making image recognition more difficult (Table 10).

It is interesting to investigate the nature of this downsampling and resulting performance comparison. Wu et al. [127] compare the tradeoff between accuracy and speed

Rank	Score	Class
1	0.2287	ant
2	0.0997	damselfly
3	0.057	nematode
4	0.0546	chainlink fence
5	0.0522	long-horned
6	0.0307	walking stick
7	0.0287	dragonfly
8	0.0267	tiger beetle
9	0.0225	doormat
10	0.0198	flute
11	0.0198	grey whale
12	0.0178	mantis
13	0.0171	lacewing
14	0.0161	radiator
15	0.0161	scabbard
16	0.0157	slide rule
17	0.0148	fly
18	0.0129	leafhopper
19	0.0101	cucumber
20	0.0094	velvet

Rank	Score	Class
1	0.103	lacewing
2	0.074	dragonfly
3	0.074	damselfly
4	0.063	walking stic
5	0.039	long-horned
6	0.027	leafhopper
7	0.025	nail
8	0.023	grasshopper
9	0.019	ant
10	0.015	mantis
11	0.015	fly
12	0.013	hammer
13	0.012	American
14	0.012	gar
15	0.011	chainlink
16	0.011	padlock
17	0.011	tree frog
18	0.011	cicada
19	0.01	screwdriver
20	0.01	harvestman

Fig. 33 Classifications of the Image to the right by different resolution models trained by Wu et al. [127]

when downsampling images to different resolutions. The researchers found that composing an ensemble of models trained with high and low-resolution images performed better than any one model individually. This ensemble prediction is found by averaging the softmax predictions. The models trained on 256×256 images and 512×512 images achieve 7.96% and 7.42% top-5 error rates, respectively. When aggregated they achieved a lower top-5 error rate of 6.97%. Therefore, different downsampled images can be viewed as another Data Augmentation scheme (Fig. 33).

With the advance of Super-Resolution Convolutional Neural Networks presented by Chong et al. [128] or SRGANs, Super-Resolution Generative Adversarial Networks, presented by Ledig et al. [129], it is interesting to consider if upsampling images to an even higher resolution would result in better models. Quality upsampling on CIFAR-10 images from even $32 \times 32 \times 3$ to $64 \times 64 \times 3$ could lead to better and more robust image classifiers.

Resolution is also a very important topic with GANs. Producing high resolution outputs from GANs is very difficult due to issues with training stability and mode collapse. Many of the newer GAN architectures such as StackGAN [130] and Progressively-Growing GANs [34] are designed to produce higher resolution images. In addition to these architectures, the use of super-resolution networks such as SRGAN could be an effective technique for improving the quality of outputs from a DCGAN [91] model. Once it is practical to produce high resolution outputs from GAN samples, these outputs will be very useful for Data Augmentation.

Final dataset size

A necessary component of Data Augmentation is the determination of the final dataset size. For example, if all images are horizontally flipped and added to the dataset, the resulting dataset size changes from N to 2N. One of the main considerations with respect to final dataset size is the additional memory and compute constraints associated with augmenting data. Practitioners have the choice between using generators which transform data on the fly during training or transforming the data beforehand and

storing it in memory. Transforming data on the fly can save memory, but will result in slower training. Storing datasets in memory can be extremely problematic depending on how heavily the dataset size has been inflated. Storing augmented datasets in memory is especially problematic when augmenting big data. This decision is generally categorized as online or offline data augmentation, (with online augmentation referring to on the fly augmentations and offline augmentation referring to editing and storing data on the disk).

In the design of a massively distributed training system, Chilimbi et al. [131] augment images before training to speed up image serving. By augmenting images in advance, the distributed system is able to request and pre-cache training batches. Augmentations can also be built into the computational graph used to construct Deep Learning models and facilitate fast differentiation. These augmentations process images immediately after the input image tensor.

Additionally, it is also interesting to explore a subset of the inflated data that will result in higher or similar performance to the entire training set. This is a similar concept to curriculum learning, since the central idea is to find an optimal ordering of training data. This idea is also very related to final dataset size and the considerations of transformation compute and available memory for storing augmented images.

Alleviating class imbalance with Data Augmentation

Class imbalance is a common problem in which a dataset is primarily composed of examples from one class. This could manifest itself in a binary classification problem such that there is a clear majority-minority class distinction, or in multi-class classification in which there is one or multiple majority classes and one or multiple minority classes. Imbalanced datasets are harmful because they bias models towards majority class predictions. Imbalanced datasets also render accuracy as a deceitful performance metric. Buda et al. [132] provide a systematic study specifically investigating the impact of imbalanced data in CNNs processing image data. Leevy et al. [27] cover many Data-level and Algorithm-level solutions to class imbalance in big data in general. Data Augmentation falls under a Data-level solution to class imbalance and there are many different strategies for implementation.

A naive solution to oversampling with Data Augmentation would be a simple random oversampling with small geometric transformations such as a 30° rotation. Other simple image manipulations such as color augmentations, mixing images, kernel filters, and random erasing can also be extended to oversample data in the same manner as geometric augmentations. This can be useful for ease of implementation and quick experimentation with different class ratios. One problem of oversampling with basic image transformations is that it could cause overfitting on the minority class which is being oversampled. The biases present in the minority class are more prevalent post-sampling with these techniques.

Oversampling methods based on Deep Learning such as adversarial training, Neural Style Transfer, GANs, and meta-learning schemes can also be used as a more intelligent oversampling strategy. Neural Style Transfer is an interesting way to create new images. These new images can be created either through extrapolating style with a foreign style or by interpolating styles amongst instances within the dataset. Using GANs

to oversample data could be another effective way to increase the minority class size while preserving the extrinsic distribution. Oversampling with GANs can be done using the entire minority class as "real" examples, or by using subsets of the minority class as inputs to GANs. The use of evolutionary sampling [133] to find these subsets to input to GANs for class sampling is a promising area for future work.

Discussion

The interesting ways to augment image data fall into two general categories: data warping and oversampling. Many of these augmentations elucidate how an image classifier can be improved, while others do not. It is easy to explain the benefit of horizontal flipping or random cropping. However, it is not clear why mixing pixels or entire images together such as in PatchShuffle regularization or SamplePairing is so effective. Additionally, it is difficult to interpret the representations learned by neural networks for GAN-based augmentation, variational auto-encoders, and meta-learning. CNN visualization has been led by Yosinski et al. [134] with their deep visualization method. Having a human-level understanding of convolutional networks features could greatly help guide the augmentation process.

Manipulating the representation power of neural networks is being used in many interesting ways to further the advancement of augmentation techniques. Traditional hand-crafted augmentation techniques such as cropping, flipping, and altering the color space are being extended with the use of GANs, Neural Style Transfer, and meta-learning search algorithms.

Image-to-image translation has many potential uses in Data Augmentation. Neural Style Transfer uses neural layers to translate images into new styles. This technique not only utilizes neural representations to separate 'style' and 'content' from images, but also uses neural transformations to transfer the style of one image into another. Neural Style Transfer is a much more powerful augmentation technique than traditional color space augmentations, but even these methods can be combined together.

An interesting characteristic of these augmentation methods is their ability to be combined together. For example, the random erasing technique can be stacked on top of any of these augmentation methods. The GAN framework possesses an intrinsic property of recursion which is very interesting. Samples taken from GANs can be augmented with traditional augmentations such as lighting filters, or even used in neural network augmentation strategies such as Smart Augmentation or Neural Augmentation to create even more samples. These samples can be fed into further GANs and dramatically increase the size of the original dataset. The extensibility of the GAN framework is amongst many reasons they are so interesting to Deep Learning researchers.

Test-time augmentation is analogous to ensemble learning in the data space. Instead of aggregating the predictions of different learning algorithms, we aggregate predictions across augmented images. We can even extend the solution algorithm to parameterize prediction weights from different augmentations. This seems like a good solution for systems concerned with achieving very high performance scores, more so than prediction

speed. Determining the effectiveness of test-time augmentation by primarily exploring test-time geometric transformations and Neural Style Transfer, is an area of future work.

An interesting question for practical Data Augmentation is how to determine post-augmented dataset size. There is no consensus as to which ratio of original to final dataset size will result in the best performing model. However, imagine using color augmentations exclusively. If the initial training dataset consists of 50 dogs and 50 cats, and each image is augmented with 100 color filters to produce 5000 dogs and 5000 cats, this dataset will be heavily biased towards the spatial characteristics of the original 50 dogs and 50 cats. This over-extensive color-augmented data will cause a deep model to overfit even worse than the original. From this anecdote, we can conceptualize the existence of an optimal size for post-augmented data.

Additionally, there is no consensus about the best strategy for combining data warping and oversampling techniques. One important consideration is the intrinsic bias in the initial, limited dataset. There are no existing augmentation techniques that can correct a dataset that has very poor diversity with respect to the testing data. All these augmentation algorithms perform best under the assumption that the training data and testing data are both drawn from the same distribution. If this is not true, it is very unlikely that these methods will be useful.

Future work

Future work in Data Augmentation will be focused on many different areas such as establishing a taxonomy of augmentation techniques, improving the quality of GAN samples, learning new ways to combine meta-learning and Data Augmentation, discovering relationships between Data Augmentation and classifier architecture, and extending these principles to other data types. We are interested in seeing how the time-series component in video data impacts the use of static image augmentation techniques. Data Augmentation is not limited to the image domain and can be useful for text, bioinformatics, tabular records, and many more.

Our future work intends to explore performance benchmarks across geometric and color space augmentations across several datasets from different image recognition tasks. These datasets will be constrained in size to test the effectiveness with respect to limited data problems. Zhang et al. [135] test their novel GAN augmentation technique on the SVHN dataset across 50, 80, 100, 200, and 500 training instances. Similar to this work, we will look to further establish benchmarks for different levels of limited data.

Improving the quality of GAN samples and testing their effectiveness on a wide range of datasets is another very important area for future work. We would like to further explore the combinatorics of GAN samples with other augmentation techniques such as applying a range of style transfers to GAN-generated samples.

Super-resolution networks through the use of SRCNNs, Super-Resolution Convolutional Neural Networks, and SRGANs are also very interesting areas for future work in Data Augmentation. We want to explore the performance differences across architectures with upsampled images such as expanding CIFAR-10 images from 32×32 to 64×64 to 128×128 and so on. One of the primary difficulties with GAN samples is trying to achieve high-resolution outputs. Therefore, it will be interesting to see how we can use super-resolution networks to achieve high-resolution such as DCGAN samples

inputted into an SRCNN or SRGAN. The result of this strategy will be compared with the performance of the Progressively Growing GAN architecture.

Test-time augmentation has the potential to make a massive difference in Computer Vision performance and has not been heavily explored. We want to establish benchmarks for different ensembles of test-time augmentations and investigate the solution algorithms used. Currently, majority voting seems to be the dominant solution algorithm for test-time augmentation. It seems highly likely that test-time augmentation can be further improved if the weight of each augmented images prediction is further parameterized and learned. Additionally, we will explore the effectiveness of test-time augmentation on object detection, comparing color space augmentations and the Neural Style Transfer algorithm.

Meta-learning GAN architectures is another exciting area of interest. Using Reinforcement Learning algorithms such as NAS on the generator and discriminator architectures seem very promising. Another interesting area of further research is to use an evolutionary approach to speed up the training of GANs through parallelization and cluster computing.

Another important area of future work for practical integration of Data Augmentation into Deep Learning workflows is the development of software tools. Similar to how the Tensorflow [136] system automates the back-end processes of gradient-descent learning, Data Augmentation libraries will automate preprocessing functions. The Keras [137] library provides an ImageDataGenerator class that greatly facilitates the implementation of geometric augmentations. Buslaev et al. presented another augmentation tool they called Albumentations [138]. The development of Neural Style Transfer, adversarial training, GANs, and meta-learning APIs will help engineers utilize the performance power of advanced Data Augmentation techniques much faster and more easily.

Conclusion

This survey presents a series of Data Augmentation solutions to the problem of overfitting in Deep Learning models due to limited data. Deep Learning models rely on big data to avoid overfitting. Artificially inflating datasets using the methods discussed in this survey achieves the benefit of big data in the limited data domain. Data Augmentation is a very useful technique for constructing better datasets. Many augmentations have been proposed which can generally be classified as either a data warping or oversampling technique.

The future of Data Augmentation is very bright. The use of search algorithms combining data warping and oversampling methods has enormous potential. The layered architecture of deep neural networks presents many opportunities for Data Augmentation. Most of the augmentations surveyed operate in the input layer. However, some are derived from hidden layer representations, and one method, DisturbLabel [28], is even manifested in the output layer. The space of intermediate representations and the label space are under-explored areas of Data Augmentation with interesting results. This survey focuses on applications for image data, although many of these techniques and concepts can be expanded to other data domains.

Data Augmentation cannot overcome all biases present in a small dataset. For example, in a dog breed classification task, if there are only bulldogs and no instances of

golden retrievers, no augmentation method discussed, from SamplePairing to AutoAugment to GANs, will create a golden retriever. However, several forms of biases such as lighting, occlusion, scale, background, and many more are preventable or at least dramatically lessened with Data Augmentation. Overfitting is generally not as much of an issue with access to big data. Data Augmentation prevents overfitting by modifying limited datasets to possess the characteristics of big data.

Abbreviations
GAN: generative adversarial network; CNN: convolutional neural network; DCGAN: deep convolutional generative adversarial network; NAS: neural architecture search; SRCNN: super-resolution convolutional neural network; SRGAN: super-resolution generative adversarial network; CT: computerized tomography; MRI: magnetic resonance imaging; PET: positron emission tomography; ROS: random oversampling; SMOTE: synthetic minority oversampling technique; RGB: red-green–blue; PCA: principal components analysis; UCI: University of California Irvine; MNIST: Modified National Institute of Standards and Technology; CIFAR: Canadian Institute for Advanced Research; t-SNE: t-distributed stochastic neighbor embedding.

Acknowledgements
We would like to thank the reviewers in the Data Mining and Machine Learning Laboratory at Florida Atlantic University. Additionally, we acknowledge partial support by the NSF (CNS-1427536). Opinions, findings, conclusions, or recommendations in this paper are solely of the authors' and do not reflect the views of the NSF.

Authors' contributions
CS performed the primary literature review and analysis for this work, and also drafted the manuscript. TMK, JLL, RAB, RZ, KW, NS, and RK worked with CS to develop the article's framework and focus. TMK introduced this topic to CS, and helped to complete and finalize this work. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Consent for publication
Not applicable.

Ethics approval and consent to participate
Not applicable.

Received: 9 January 2019 Accepted: 22 April 2019
Published online: 06 July 2019

References

1. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Adv Neural Inf Process Syst. 2012;25:1106–14.
2. Karen S, Andrew Z. Very deep convolutional networks for large-scale image recognition. arXiv e-prints. 2014.
3. Kaiming H, Xiangyu Z, Shaoqing R, Jian S. Deep residual learning for image recognition. In: CVPR, 2016.
4. Christian S, Vincent V, Sergey I, Jon S, Zbigniew W. Rethinking the inception architecture for computer vision. arXiv e-prints, 2015.
5. Gao H, Zhuang L, Laurens M, Kilian QW. Densely connected convolutional networks. arXiv preprint. 2016.
6. Jan K, Vladimir G, Daniel C. Regularization for deep learning: a taxonomy. arXiv preprint. 2017.
7. Nitish S, Geoffrey H, Alex K, Ilya S, Ruslan S. Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res. 2014;15(1):1929–58.
8. Jonathan T, Ross G, Arjun J, Yann L, Christoph B. Efficient object localization using convolutional networks. In: CVPR'15. 2015.
9. Sergey I, Christan S. Batch normalization: accelerating deep network training by reducing internal covariate shift. In: ICML; 2015.
10. Karl W, Taghi MK, DingDing W. A survey of transfer learning. J Big Data. 2016;3:9.
11. Shao L. Transfer learning for visual categorization: a survey. IEEE Trans Neural Netw Learn Syst. 2015;26(5):1019–34.
12. Jia D, Wei D, Richard S, Li-Jia L, Kai L, Li F-F. ImageNet: a large-scale hierarchical image database. In: CVPR09, 2009.
13. Amir Z, Alexander S, William S, Leonidas G, Jitendra M, Silvio S. Taskonomy: disentangling task transfer learning. In: CVPR'18. 2018.
14. Yosinski J, Clune J, Bengio Y, Lipson H. How transferable are features in deep neural networks? Adv Neural Inf Process Syst. 2014;27:3320–8.

15. Erhan D, Bengio Y, Courville A, Manzagol PA, Vincent P. Why does unsupervised pre-training help deep learning? J Mach Learn Res. 2010;11:625–60.
16. Mark P, Dean P, Geoffrey H, Tom MM. Zero-shot learning with semantic output codes. In: NIPS; 2009.
17. Yongqin X, Christoph HL, Bernt S, Zeynep A. Zero-shot learning—a comprehensive evaluation of the good, the bad and the ugly. arXiv preprint. 2018.
18. Yaniv T, Ming Y, Marc' AR, Lior W. DeepFace: closing the gap to human-level performance in face verification. In: CVPR '14; 2014.
19. Gregory K, Richard Z, Ruslan S. Siamese neural networks for one-shot image recognition. In: ICML Deep Learning workshop; 2015.
20. Adam S, Sergey B, Matthew B, Dean W, Timothy L. One-shot learning with memory-augmented neural networks. arXiv preprint. 2016.
21. Tomas M, Ilya S, Kai C, Greg C, Jeffrey D. Distributed representations of words and phrases and their compositionality. Accepted to NIPS 2013.
22. Jeffrey P, Richard S, Christopher DM. GloVe: global vectors for word representation. In: Proceedings of the empirical methods in natural language processing (EMNLP 2014) 12. 2014.
23. Halevy A, Norvig P, Pereira F. The unreasonable effectiveness of data. IEEE Intell Syst. 2009;24:8–12.
24. Chen S, Abhinav S, Saurabh S, Abhinav G. Revisting unreasonable effectivness of data in deep learning era. In: ICCV; 2017. p. 843–52.
25. Esteva A, Kuprel B, Novoa RA, Ko J, Swetter SM, Blau HM, Thrun S. Dermatologist-level classification of skin cancer with deep neural networks. Nature. 2017;542:115–8.
26. Geert L, Thijs K, Babak EB, Arnaud AAS, Francesco C, Mohsen G, Jeroen AWM, van Bram G, Clara IS. A survey on deep learning in medical image analysis. Med Image Anal. 2017;42:60–88.
27. Joffrey LL, Taghi MK, Richard AB, Naeem S. A survey on addressing high-class imbalance in big data. Springer J Big Data. 2018;5:42.
28. LeCun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proc IEEE. 1998;86(11):2278–324.
29. Nitesh VC, Kevin WB, Lawrence OH, Kegelmeyer W. SMOTE: synthetic minority over-sampling technique. J Artif Intellig Res. 2002;16:321–57.
30. Hui H, Wen-Yuan W, Bing-Huan M. Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Proceedings of ICIC, vol. 3644, Lecture Notes in Computer Science, New York. 2005, p. 878–87.
31. Ian JG, Jean PA, Mehdi M, Bing X, David WF, Sherjil O, Aaron C, Yoshua B. Generative adversarial nets. NIPS. 2014.
32. Leon AG, Alexander SE, Matthias B. A neural algorithm of artistic style. ArXiv. 2015.
33. Barret Z, Quoc VL. Neural architecture search with reinforcement learning. In: International conference on learning representatoins, 2017.
34. Tero K, Timo A, Samuli L, Jaakko L. Progressive growing of GANs for improved quality, stability, and variation. CoRR, abs/1710.10196, 2017.
35. Justin J, Alexandre A, Li FF. Perceptual losses for real-time style transfer and super-resolution. ECCV. 2016;2016:694–711.
36. Luis P, Jason W. The effectiveness of data augmentation in image classification using deep learning. In: Stanford University research report, 2017.
37. Lemley J, Barzrafkan S, Corcoran P. Smart augmentation learning an optimal data augmentation strategy. In: IEEE Access. 2017.
38. Ekin DC, Barret Z, Dandelion M, Vijay V, Quoc VL. AutoAugment: learning augmentation policies from data. ArXiv preprint. 2018.
39. Xin Y, Paul SB, Ekta W. Generative adversarial network in medical imaging: a review. arXiv preprint. 2018.
40. Jelmer MW, Tim L, Max AV, Ivana I. Generative adversarial networks for noise reduction in low-dose CT. In: IEEE Transactions on Medical Imaging. 2017.
41. Ohad S, Tammy RR. Accelerated magnetic resonance imaging by adversarial neural network. In: DLMIA/ML-CDS@ MICCAI, 2017.
42. Wang Y, Biting Y, Wang L, Chen Z, Lalush DS, Lin W, Xi W, Zhou J, Shen D, Zhou L. 3D conditional generative adversarial networks for high-quality PET image estimation at low dose. NeuroImage. 2018;174:550–62.
43. Dwarikanath M, Behzad B. Retinal vasculature segmentation using local saliency maps and generative adversarial networks for image super resolution. arXiv preprint. 2017.
44. Francesco C, Aldo M, Claudio S, Giorgio T. Biomedical data augmentation using generative adversarial neural networks. In: International conference on artificial neural networks. Berlin: Springer; 2017. P. 626–34.
45. Camilo B, Andrew JP, Larry TD, Allen TN, Susan MR, Bennett AL. Learning implicit brain MRI manifolds with deep learning. Int Soc Opt Photonics. 2018;10574:105741.
46. Maria JMC, Sarfaraz H, Jeremy B, Ulas B. How to fool radiologists with generative adversarial networks? A visual turing test for lung cancer diagnosis. arXiv preprint. 2017.
47. Baur C, Albarqouni S, Navab N. MelanoGANs: high resolution skin lesion synthesis with GANs. arXiv preprint. 2018.
48. Madani A, Moradi M, Karargyris A, Syeda-Mahmood T. Chest x-ray generation and data augmentation for cardiovascular abnormality classification. In: Medical imaging 2018. Image Processing 2018;10574:105741.
49. Maayan F-A, Eyal K, Jacob G, Hayit G. GAN-based data augmentation for improved liver lesion classification. arXiv preprint. 2018.
50. Joseph R, Santosh D, Ross G, Ali F. You only look once: unified, real-time object detection. In: CVPR'16. 2016.
51. Ross G, Jeff D, Trevor D, Jitendra M. Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR '14. 2014.
52. Ross G. Fast R-CNN. CoRR, abs/1504.08083. 2015.
53. Shaoqing R, Kaiming H, Ross G, Jian S. Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS, 2015.
54. Jonathan L, Evan S, Trevor D. Fully convolutional networks for semantic segmentation. CoRR, abs/1411.4038. 2014.

55. Olaf R, Philipp F, Thomas B. U-Net: convolutional networks for biomedical image segmentation. In: MICCAI. Springer; 2015, p. 234–41.
56. Hessam B, Maxwell H, Mohammad R, Ali F. Label refinery: improving imagenet classification through label progression. arXiv preprint. 2018.
57. Francisco JM-B, Fiammetta S, Jose MJ, Daniel U, Leonardo F. Forward noise adjustment scheme for data augmentation. arXiv preprints. 2018.
58. Dua D, Karra TE. UCI machine learning repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science; 2017.
59. Ken C, Karen S, Andrea V, Andrew Z. Return of the devil in the details: delving deep into convolutional nets. In: Proceedings of BMVC. 2014.
60. Mark E, Luc VG, Christopher KIW, John W, Andrew Z. The pascal visual object classes (VOC) challenge. http://www.pascal-network.org/challenges/VOC/voc2008/workshop/. 2008.
61. Aranzazu J, Miguel P, Mikel G, Carlos L-M, Daniel P. A comparison study of different color spaces in clustering based image segmentation. IPMU; 2010.
62. Quanzeng Y, Jiebo L, Hailin J, Jianchao Y. Robust image sentiment analysis using progressively trained and domain transferred deep networks. In: AAAI. 2015, p. 381–8.
63. Luke T, Geoff N. Improving deep learning using generic data augmentation. arXiv preprint. 2017.
64. Guoliang K, Xuanyi D, Liang Z, Yi Y. PatchShuffle regularization. arXiv preprint. 2017.
65. Hiroshi I. Data augmentation by pairing samples for images classification. ArXiv e-prints. 2018.
66. Cecilia S, Michael JD. Improved mixed-example data augmentation. ArXiv preprint. 2018.
67. Daojun L, Feng Y, Tian Z, Peter Y. Understanding mixup training methods. In: IEEE access. 2018. p. 1.
68. Ryo T, Takashi M. Data augmentation using random image cropping and patches for deep CNNs. arXiv preprints. 2018.
69. Yoshua B, Jerome L, Ronan C, Jason W. Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning, ACM. 2009, p. 41–8.
70. Zhun Z, Liang Z, Guoliang K, Shaozi L, Yi Y. Random erasing data augmentation. ArXiv e-prints. 2017.
71. Terrance V, Graham WT. Improved regularization of convolutional neural networks with cutout. arXiv preprint. 2017.
72. Agnieszka M, Michal G. Data augmentation for improving deep learning in image classification problem. In: IEEE 2018 international interdisciplinary Ph.D. Workshop, 2018.
73. Jonathan K, Michael S, Jia D, Li F-F. 3D object representations for fine-grained categorization. In: 4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013 (3dRR-13). Sydney, Australia. Dec. 8, 2013.
74. Tomohiko K, Michiaki I. Icing on the cake: an easy and quick post-learning method you can try after deep learning. arXiv preprints. 2018.
75. Terrance V, Graham WT. Dataset augmentation in feature space. In: Proceedings of the international conference on machine learning (ICML), workshop track, 2017.
76. Sebastien CW, Adam G, Victor S, Mark DM. Understanding data augmentation for classification: when to warp? CoRR, abs/1609.08764, 2016.
77. Seyed-Mohsen MD, Alhussein F, Pascal F. DeepFool: a simple and accurate method to fool deep neural networks. arXiv preprint. 2016.
78. Jiawei S, Danilo VV, Sakurai K. One pixel attack for fooling deep neural networoks. arXiv preprints. 2018.
79. Michal Z, Konrad Z, Negar R, Pedro OP. Adversarial framing for image and video classification. arXiv preprints. 2018.
80. Logan E, Brandon T, Dimitris T, Ludwig S, Aleksander M. A rotation and a translation suffice: fooling CNNs with simple transformations. ArXiv preprint. 2018.
81. Goodfellow I, Shlens J, Szegedy C. Explaining and Harnessing Adversarial Examples. International Conference on Learning Representations, 2015.
82. Ian JG, David W-F, Mehdi M, Aaron C, Yoshua B. Maxout networks. arXiv preprint. 2013.
83. Shuangtao L, Yuanke C, Yanlin P, Lin B. Learning more robust features with adversarial training. ArXiv preprints. 2018.
84. Lingxi X, Jingdong W, Zhen W, Meng W, Qi T. DisturbLabel: regularizing CNN on the loss layer. arXiv preprint. 2016.
85. Christopher B, Liang C, Ricardo GPB, Roger G, Alexander H, David AD, Maria VH, Joanna W, Daniel R. GAN augmentation: augmenting training data using generative adversarial networks. arXiv preprint. 2018.
86. Doersch C. Tutorial on Variational Autoencoders. ArXiv e-prints. 2016.
87. Laurens M, Geoffrey H. Visualizing data using t-SNE. J Mach Learn Res. 2008;9:2431–56.
88. Jeff D, Philipp K, Trevor D. Adversarial feature learning. In: CVPR'16. 2016.
89. Lin Z, Shi Y, Xue Z. IDSGAN: Generative Adversarial Networks for Attack Generation against Intrusion Detection. arXiv preprint; 2018.
90. William F, Mihaela R, Balaji L, Andrew MD, Shakir M, Ian G. Many paths to equilibrium: GANs do not need to decrease a divergence at every step. In: International conference on learning representations (ICLR); 2017.
91. Alec R, Luke M, Soumith C. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR, 2016.
92. Jun-Yan Z, Taesung P, Phillip I, Alexei AE. Unpaired image-to-image translation using cycle-consistent adversarial networks. In: International conference on cmoputer vision (ICCV), 2017.
93. Xinyue Z, Yifan L, Zengchang Q, Jiahong L. Emotion classification with data augmentation using generative adversarial networks. CoRR, vol. abs/1711.00648. 2017.
94. Goodfellow IJ, Erhan D, Carrier PL, Courville A, Mirza M, Hamner B, Cukierski W, Tang Y, Thaler D, Lee DH, et al. Challenges in representation learning: A report on three machine learning contests. In: NIPS. Berlin: Springer; 2013. p.117–24.
95. Mehdi M, Simon O. Conditional generative adversarial nets. arXiv preprint. 2014.
96. Mario L, Karol K, Marcin M, Olivier B, Sylvain G. Are GANs created equal? A large-scale study. arXiv preprint. 2018.

97. Swee KL, Yi L, Ngoc-Trung T, Ngai-Man C, Gemma R, Yuval E. DOPING: generative data augmentation for unsupervised anomaly detection with GAN. arXiv preprint. 2018.
98. Alireza M, Jonathon S, Navdeep J, Ian G, Brendan F. Adversarial autoencoders. arXiv preprint. 2015.
99. Tim S, Ian G, Wojciech Z, Vicki C, Alec R, Xi C. Improved techniques for training GANs. arXiv preprint. 2016.
100. Yanghao L, Naiyan W, Jiaying L, Xiaodi H. Demistifying neural style transfer. arXiv preprint. 2017.
101. Khizar H. Super-resolution via deep learning. arXiv preprint. 2017.
102. Dmitry U, Andrea V, Victor L. Instance normalization: the missing ingredient for fast stylization. arXiv preprint. 2016.
103. Philip TJ, Amir AA, Stephen B, Toby B, Boguslaw O. Style augmentation: data augmentation via style randomization. arXiv e-prints. 2018.
104. Josh T, Rachel F, Alex R, Jonas S, Wojciech Z, Pieter A. Domain randomization for transferring deep neural networks from simulation to the real world. arXiv preprint. 2017.
105. Ashish S, Tomas P, Oncel T, Josh S, Wenda W, Russ W. Learning from simulated and unsupervised images through adversarial training. In: Conference on computer vision and pattern recognition, 2017.
106. Stephan RR, Vibhav V, Stefan R, Vladlen K. Playing for data: ground truth from computer games. In: European conference on computer vision (ECCV); 2016.
107. Brostow Gabriel J, Fauqueur Julien, Cipolla Roberto. Semantic object classes in video: a high-definition ground truth database. Pattern Recogn Lett. 2008;30(2):88–97.
108. Marius C, Mohamed O, Sebastian R, Timo R, Markus E, Rodrigo B, Uwe F, Stefan R, Bernt S. The cityscape dataset for semantic urban scene understanding. In: CVPR; 2016.
109. Esteban R, Sherry M, Andrew S, Saurabh S, Yutaka LS, Jie T, Quoc VL, Alexey K. Large-scale evolution of image classifiers. In: Proceedings of the 34th international conference on machine learning (ICML '17). 2017.
110. Esteban R, Alok A, Yanping H, Quoc VL. Regularized evolution for image classifier architecture search. arXiv preprint. 2018.
111. Tim S, Jonathan H, Xi C, Szymon S, Ilya S. Evolution strategies as a scalable alternative to reinforcement learning. arXiv e-prints. 2017.
112. Horia M, Aurelia G, Benjamin R. Simple random search provides a competitive approach to reinforcement learning. In: Advances in neural information processing systems (NIPS); 2018.
113. David GL. Distinctive image features from scale-invariant keypoints. Int J Comput Vis. 2004;2004:91–110.
114. Navneet D, Bill T. Histograms of oriented gradients for human detection. In: CVPR, 2005.
115. Sutton RS, Reinforcement AG. Learning: an introduction. New York: MIT Press; 1998.
116. Mingyang G, Kele X, Bo D, Huaimin W, Lei Z. Learning data augmentation policies using augmented random search. arXiv preprint. 2018.
117. Tran NM, Mathieu S, Hoang TL, Martin W. Automated image data preprocessing with deep reinforcement learning. arXiv preprints. 2018.
118. Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int J Uncertain Fuzzin Know Based Syst. 1998;6(02):107–16.
119. Jia S, Wang P, Jia P, Hu S. Research on data augmentation for image classification based on convolutional neural networks. In: 2017 Chinese automation congress (CAC), 2017. p. 4165–70.
120. Ilija R, Piotr D, Ross G, Georgia G, Kaiming H. Data distillation: towards omni-supervised learning. In: CVPR '18; 2018.
121. Guotai W, Michael A, Sebastien O, Wenqi L, Jan D, Tom V. Test-time augmentation with uncertainty estimation for deep learning-based medical image segmentation. OpenReview.net. 2018.
122. Fabio P, Christina V, Sandra A, Eduardo V. Data augmentation for skin lesion analysis. In: ISIC skin image analysis workshop and challenge @ MICCAI 2018. 2018.
123. Karzuhisa M, Akira H, Akane M, Hiroshi K. Image classification of melanoma, nevus and seborrheic keratosis by deep neural network ensemble. In: International skin imaging collaboration (ISIC) 2017 challenge at the international symposium on biomedical imaging (ISBI). 2017.
124. Max J, Karen S, Andrea V, Andrew Z. Synthetic data and artificial neural networks for natural scene text recognition. arXiv preprint. 2014.
125. Florian S, Dmitry K, James P. FaceNet: a unified embedding for face recognition and clustering. In: CVPR'15. 2015.
126. Xudong M, Qing L, Haoran X, Raymond YKL, Zhen W, Stephen PS. Least squares generative adversarial networks. In: International conference on computer vision (ICCV), 2017.
127. Ren W, Shengen Y, Yi S, Qingqing D, Gang S. Deep image: scaling up image recognition. CoRR, abs/1501.02876, 2015.
128. Chao D, Chen CL, Kaiming H, Ziaoou T. Learning a deep convolutional network for image super-resolution. In: ECCV. Berlin: Springer; 2014. , p. 184–99.
129. Christian L, Lucas T, Ferenc H, Jose C, Andrew C, Alejandro A, Andrew A, Alykhan T, Johannes T, Zehan W, Wenzhe S. Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint. 2016.
130. Han Z, Tao X, Hongsheng L, Shaoting Z, Xiaogang W, Xiaolei H, Dimitris M. StackGAN: text to photo-realistic image synthesis with stacked generative adversarial networks. In: ICCV, 2017.
131. Trishul C, Yutaka S, Johnson A, Karthik K. Project adam: building an efficient and scalable deep learning training system. In: Proceedings of OSDI. 2014. P. 571–82.
132. Buda Mateusz, Maki Atsuto, Mazurowski Maciej A. A systematic study of the class imbalance problem in convolutional neural networks. Neural Networks. 2018;106:249–59.
133. Drown DJ, Khoshgoftaar TM, Seliya N. Evolutionary sampling and software quality modeling of high-assurance systems. IEEE Trans Syst. 2009;39(5):1097–107.
134. Jason Y, Jeff C, Anh N, Thomas F, Hod L. Understanding neural networks through deep visualization. In: European conference on computer vision (ECCV). Berlin: Springer; 2015. p. 818–33.
135. Xiaofeng Z, Zhangyang W, Dong L, Qing L. DADA: deep adversarial data augmentation for extremely low data regime classification. arXiv preprint. 2018.
136. Martin A, Paul B, Jianmin C, Zhifeng C, Andy D, Jeffrey D, Matthieu D, Sanjay G, Geoffrey I, Michael I, Manjunath K, Josh L, Rajat M, Sherry M, Derek GM, Benoit S, Paul T, Vijay V, Pete W, Matrin W, Yuan Y, Xiaoqiang Z. TensorFlow:

a system for large-scale machine learning. In: Proceedings of the 12th USENIX symposium on operating system design and implementation (OSDI '16), 2016.
137. Keras https://keras.io/. 2015.
138. Alexander B, Alex P, Eugene K, Vladimir II, Alexandr AK. Albumentations: fast and flexible image augmentations. ArXiv preprints. 2018.
139. Maayan F-A, Idit D, Eyal K, Michal A, Jacob G, Hayit G. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. arXiv preprint. 2018.
140. Changhee H, Hideaki H, Leonardo R, Ryosuke A, Wataru S, Shinichi M, Yujiro F, Giancarlo M, Hideki N. GAN-based synthetic brain mr image generation. In: 2018 IEEE 15th International Symposium on biomedical imaging (ISBI 2018). IEEE, 2011. P. 734–8.

Publisher's Note