Smart Augmentation Learning an Optimal Data Augmentation Strategy

JOSEPH LEMLEY, (Student Member, IEEE), SHABAB BAZRAFKAN, (Student Member, IEEE), AND PETER CORCORAN, (Fellow, IEEE)

Collage of Engineering and Informatics, National University of Ireland Galway, SW4 794 Galway, Ireland

Corresponding author: Joseph Lemley (j.lemley2@nuigalway.ie)

ABSTRACT A recurring problem faced when training neural networks is that there is typically not enough data to maximize the generalization capability of deep neural networks. There are many techniques to address this, including data augmentation, dropout, and transfer learning. In this paper, we introduce an additional method, which we call smart augmentation and we show how to use it to increase the accuracy and reduce over fitting on a target network. Smart augmentation works, by creating a network that learns how to generate augmented data during the training process of a target network in a way that reduces that networks loss. This allows us to learn augmentations that minimize the error of that network. Smart augmentation has shown the potential to increase accuracy by demonstrably significant measures on all data sets tested. In addition, it has shown potential to achieve similar or improved performance levels with significantly smaller network sizes in a number of tested cases.

INDEX TERMS Artificial intelligence, artificial neural networks, machine learning, computer vision supervised learning, machine learning algorithms, image databases.

I. INTRODUCTION

In order to train a deep neural network, the first and probably most important task is to have access to enough labeled samples of data. Not having enough quality labeled data will generate overfitting, which means that the network is highly biased to the data it has seen in the training set and, therefore will not be able to generalize the learned model to any other samples. In [1] there is a discussion about how much the diversity in training data and mixing different datasets can affect the model generalization. Mixing several datasets might be a good solution, but it is not always feasible due to lack of accessibility.

One of the other approaches to solving this problem is using different regularization techniques. In recent years different regularization approaches have been proposed and successfully tested on deep neural network models. The dropout technique [2] and batch normalization [3] are two well-known regularization methods used to avoid overfitting when training deep models.

Another technique for addressing this problem is called augmentation. Data augmentation is the process of supplementing a dataset with similar data that is created from the information in that dataset. The use of augmentation in deep learning is ubiquitous, and when dealing with images, often includes the application of rotation, translation, blurring and other modifications to existing images that allow a network to better generalize [4].

Augmentation serves as a type of regularization, reducing the chance of overfitting by extracting more general information from the database and passing it to the network. One can classify the augmentation methods into two different types. The first is unsupervised augmentation. In this type of augmentation, the data expansion task is done regardless of the label of the sample. For example adding a different kind of noise, rotating or flipping the data. These kinds of data augmentations are usually not difficult to implement.

One of the most challenging kinds of data expansion is mixing different samples with the same label in feature space in order to generate a new sample with the same label. The generated sample has to be recognizable as a valid data sample, and also as a sample representative of that specific class. Since the label of the data is used to generate the new sample, this kind of augmentation this can be viewed as a type of supervised augmentation.

Many deep learning frameworks can generate augmented data. For example, Keras [5] has a built in method to randomly flip, rotate, and scale images during training but not all of these methods will improve performance and should not be

used "blindly". For example, on MNIST (The famous handwritten number dataset), if one adds rotation, the network will be unable to distinguish properly between handwritten "6" and "9" digits. Likewise, a system that uses deep learning to classify or interpret road signs may become incapable of discerning left and right arrows if the training set was augmented with by indiscriminate flipping of images.

More sophisticated types of augmentation, such as selectively blending images or adding directional lighting rely on expert knowledge. Besides intuition and experience, there is no universal method that can determine if any specific augmentation strategy will improve results until after training. Since training deep neural nets is a time-consuming process, this means only a limited number of augmentation strategies will likely be attempted before deployment of a model.

Blending several samples in the dataset in order to highlight their mutual information is not a trivial task in practice. Which samples should be mixed together how many of them and how they mixed is a big problem in data augmentation using blending techniques.

Augmentation is typically performed by trial and error, and the types of augmentation performed are limited to the imagination, time, and experience of the researcher. Often, the choice of augmentation strategy can be more important than the type of network architecture used [6].

Before Convolutional Neural Networks (CNN) became the norm for computer vision research, features were "handcrafted". Handcrafting features went out of style after it was shown that Convolutional Neural Networks could learn the best features for a given task. We suggest that since the CNN can generate the best features for some specific pattern recognition tasks, it might be able to give the best feature space in order to merge several samples in a specific class and generate a new sample with the same label. Our idea is to generate the merged data in a way that produces the best results for a specific target network through the intelligent blending of features between 2 or more samples.

II. RELATED WORK

Manual augmentation techniques such as rotating, flipping and adding different kinds of noise to the data samples, are described in depth in [4] and [7] which attempt to measure the performance gain given by specific augmentation techniques. They also provide a list of recommended data augmentation methods.

In 2014, Srivastava *et al.* introduced the dropout technique [2] aiming to reduce overfitting, especially in cases where there is not enough data. Dropout works by temporarily removing a unit (or artificial neuron) from the Artificial Neural Network and any connections to or from that unit.

Konda *et al.* Showed that dropout can be used for data augmentation by "projecting the the dropout noise within a network back into the input space" [8].

Jaderberg *et al.* devised an image blending strategy as part of their paper "Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition" [9]. They

used what they call "natural data blending" where each of the image layers is blended with a randomly sampled crop of an image from a training dataset. They note a significant (+44%) increase in accuracy using such synthetic images when image layers are blended together via a random process.

Another related technique is training on adversarial examples. Goodfellow *et al.* note that, although augmentation is usually done with the goal of creating images that are as similar as possible to the natural images one expects in the testing set, this does not need to be the case. They further demonstrate that training with adversarial examples can increase the generalization capacity of a network, helping to expose and overcome flaws in the decision function [10].

The use of Generative Adversarial Neural Networks [11] is a very powerful unsupervised learning technique that uses a min-max strategy wherein a 'counterfeiter' network attempts to generate images that look enough like images within a dataset to 'fool' a second network while the second network learns to detect counterfeits. This process continues until the synthetic data is nearly indistinguishable from what one would expect real data to look like. Generative Adversarial Neural Networks can also be used to generate images that augment datasets, as in the strategy employed by Shrivastav *et al.* [12].

Another method of increasing the generalization capacity of a neural network is called "transfer learning". In transfer learning, we want to take knowledge learned from one network, and transfer it to another [13]. In the case of Convolutional Neural Networks, when used as a technique to reduce overfitting due to small datasets, it is common to use the trained weights from a large network that was trained for a specific task and to use it as a starting point for training the network to perform well on another task.

Batch normalization, introduced in 2015, is another powerful technique. It was discovered upon the realization that normalization need not just be performed on the input layer, but can also be achieved on intermediate layers [3].

Like the above regularization methods, Smart Augmentation attempts to address the issue of limited training data to improve regularization and reduce overfitting. As with [10], our method does not attempt to produce augmentations that appear "natural". Instead, our network learns to combine images in ways that improve regularization. Unlike [4] and [7], we do not address manual augmentation, nor does our network attempt to learn simple transformations. Unlike the approach of image blending in [9], we do not arbitrarily or randomly blend images. Smart augmentation can be used in conjunction with other regularization techniques, including dropout and traditional augmentation.

III. SMART AUGMENTATION

Smart Augmentation is the process of learning suitable augmentations when training deep neural networks.

The goal of Smart Augmentation is to learn the best augmentation strategy for a given class of input data. It does this by learning to merge two or more samples
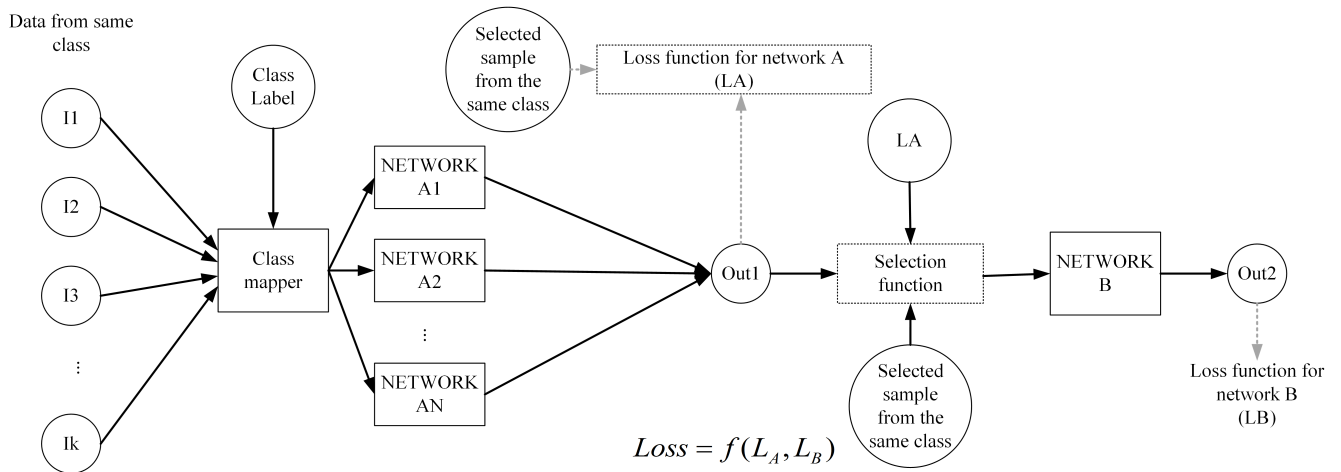
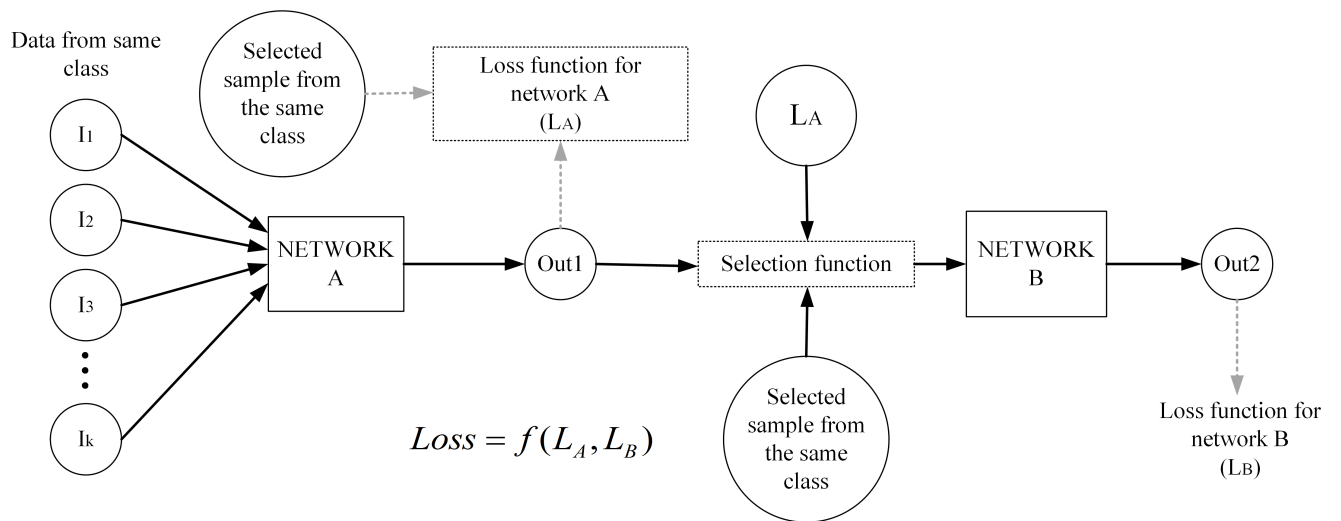**FIGURE 1.** Smart augmentation with more than one network A.

FIGURE 2. Diagram illustrating the reduced smart augmentation concept with just one network A.

in one class. This merged sample is then used to train a target network. The loss of the target network is used to inform the augmenter at the same time. This has the result of generating more data for use by the target network. This process often includes letting the network come up with unusual or unexpected but highly performant augmentation strategies.

A. TRAINING STRATEGY FOR SMART AUGMENTATION
During the training phase, we have two networks: Network A, which generates data; and network B, which is the network that will perform a desired task (such as classification). The main goal is to train network B to do some specific task while there are not enough representative samples in the given dataset. To do so, we use another network A to generate new samples.

This network accepts several inputs from the same class (the sample selection could be random, or it could use

some form of clustering, either in the pixel space or in the feature space) and generates an output which approximates data from that class. This is done by minimizing the loss function LA which accepts out1 and image i as input. Where out1 is the output of network A and mage i is a selected sample from the same class as the input. The only constraint on the network A is that the input and output of this network should be the same shape and type. For example, if N samples of a P-channel image are fed to network A, the output will be a single P-channel image.

B. THE GENERATIVE NETWORK A AND LOSS FUNCTION
The loss function can be further parameterized by the inclusion of α and β as $f(L_A, L_B; \alpha, \beta)$. In the experiments and results sections of this paper, we examine how these can impact final accuracy.

Network A can either be implemented as a single network (figure 2) or as multiple networks, as in figure 1. Using

more than one network A has the advantage that the networks can learn class-specific augmentations that may not be suitable for other classes, but which work well for the given class.

Network A is a neural network, such as a generative model, with the difference that network A is being influenced by network B in the back propagation step, and network A accepts multiple samples as input simultaneously instead of just one at a time. This causes the data generated by network A to converge to the best choices to train network B for that specific task, and at the same time, it is controlled by loss function LA in a way that ensures that the outputs are similar to other members of its class.

The overall loss function during training is $f(LA, LB)$ where f is a function whose output is a transformation of LA and LB. This function could be an epoch-dependent function i.e. the function could change with the epoch number. In the training process, the error back-propagates from network B to network A. This tunes network A to generate the best augmentations for network B. After training is finished, Network A is cut out of the model and network B is used in the test process. The joint information between data samples is exploited to both reduce overfitting and to increase the accuracy of the target network during training.
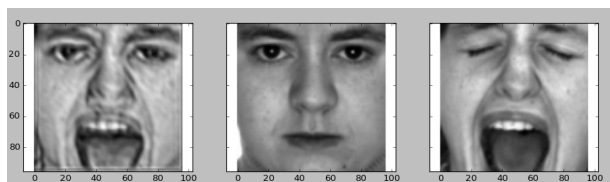

FIGURE 3. The image on the left is created by a learned combination of the two images on the right. This type of image transformation helped increase the accuracy of network B. The image was not produced to be an ideal approximation of a face but instead, contains features that helped network B better generalize the concept of gender which is the task it was trained for.

C. HOW SMART AUGMENTATION WORKS

The proposed method uses a network (network A) to learn the best sample blending for the specific problem. The output of network A is the used for the input of network B. The idea is to use network A to learn the best data augmentation to train network B. Network A accepts several samples from the same class in the dataset and generates a new sample from that class, and this new sample should reduce the training loss for network B. In figure 3 we see an output of network A designed to do the gender classification. The image on the left is a merged image of the other two. This image represents a sample from the class ''male'' that does not appear in the dataset, but still, has the identifying features of its class.

Notice that in figure 3, an image was created with an open mouth and open eyes from two images. The quality of the face image produced by network A does not matter. Only its ability to help network B better generalize. Our approach is most applicable to classification tasks but may also have applications in any approach where the selective blending of sample

features improves performance. Our observations show that this approach can reduce overfitting and increase accuracy. In the following sections, we evaluate several implementations of our smart augmentation technique on various datasets to show how it can improve accuracy and prevent overfitting. We also show that with smart augmentation, we can train a very small network to perform as well as (or better than) a much larger network that produces state of the art results.

IV. METHODS

Experiments were conducted on NVIDIA Titan X GPU's running a pascal architecture with python 2.7, using the Theano [14] and Lasange frameworks.

A. DATA PREPARATION

To evaluate our method, we chose 4 datasets with characteristics that would allow us to examine the performance of the algorithm on specific types of data. Since the goal of our paper is to measure the impact of the proposed technique, we do not attempt to provide a comparison of techniques that work well on these databases. For such a comparison we refer to [15] for gender datasets or [16] for the places dataset.

1) HIGHLY CONSTRAINED FACES DATASET (db1)

Our first dataset, db1 was composed from the AR faces database [17] with a total of 4,000 frontal faces of male and female subjects. The data was split into subject exclusive training, validation, and testing sets, with 70% for training, 20% for validation, and 10% for testing. All face images were reduced to 96X96 grayscale with pixel values normalized between 0 and 1.

2) AUGMENTED, HIGHLY CONSTRAINED
FACES DATASET (db1a)

To compare traditional augmentation with smart augmentation and to examine the effect of traditional augmentation on smart augmentation, we created an augmented version of db1 with every combination of flipping, blurring, and rotation (-5,-2,0,2,5 degrees with the axis of rotation at the center of the image). This resulted in a larger training set of 48360 images. The test and validation sets were unaltered from db1. The data was split into a subject exclusive training, validation, and testing sets with 70% for training, 20% for validation, and 10% for testing. All face images were reduced to 96X96 with pixel values normalized between 0 and 1.

3) FERET

Our second dataset, db2, was the FERET dataset. We converted FERET to grayscale and reduced the size of each image to 100X100 with pixel values normalized between 0 and 1. The data was split into subject exclusive training, validation, and testing sets, with 70% for training, 20% for validation and 10% for testing.

Color FERET [18] Version 2 was collected between December 1993 and August 1996 and made freely available with the intent of promoting the development of face

FIGURE 4. Arbitrarily selected images from FERET demonstrate similarities in lighting, pose, subject, background, and other photographic conditions.
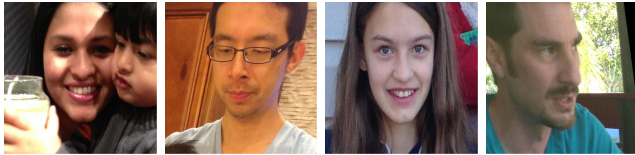

FIGURE 5. Arbitrarily selected images from the Adience show significant variations in lighting, pose, subject, background, and other photographic conditions.

recognition algorithms. The images are labeled with gender, pose, and name.

Although FERET contains a large number of high-quality images in different poses and with varying face obstructions (beards, glasses, etc), they all have certain similarities in quality, background, pose, and lighting that make them very easy for modern machine learning methods to correctly classify (see figure 4). In our experiments, we use all images in FERET for which gender labels exist.

4) ADIENCE

Our third dataset, db3, was Adience (see figure 5). We converted Adience to grayscale images with size 100×100 and normalized the pixel values between 0 and 1. The data was split into subject exclusive training, validation, and testing sets, with 70% for training, 20% for validation and 10% for testing.

5) DB4

Our fourth dataset, db4, was the MIT places dataset [16]. The MIT PLACES dataset is a machine learning database containing has 205 scene categories and 2.5 million labeled images.

The Places Dataset is unconstrained and includes complex scenery in a variety of lighting conditions and environments, as shown in figure 6.

We restricted ourselves to just the first two classes in the dataset (Abbey and Airport). Pixel values were normalized between 0 and 1. The "small dataset," which had been rescaled to 256×256 with 3 color channels, was used for all experiments without modification except for normalization of the pixel values between 0 and 1.

V. EXPERIMENTS

In these experiments, we call network B the network that is being trained for a specific task (such as classification).

FIGURE 6. Example images from the MIT places dataset showing two examples from each of the two classes (abbey and airport) used in our experiments.

TABLE 1. Full listing of experiments.

Exp	DB	Net A	A ch	Net B	α	β	LR	Momentum
1	db1	1	1	B_1	0.3	0.7	0.01	0.9
2	db1	1	2	B_1	0.3	0.7	0.01	0.9
3	db1	1	3	B_1	0.3	0.7	0.01	0.9
4	db1	1	4	B_1	0.3	0.7	0.01	0.9
5	db1	1	5	B_1	0.3	0.7	0.01	0.9
6	db1	1	6	B_1	0.3	0.7	0.01	0.9
7	db1	1	7	B_1	0.3	0.7	0.01	0.9
8	db1	1	8	B_1	0.3	0.7	0.01	0.9
9	db1	2	1	B_1	0.3	0.7	0.005	0.9
10	db1	2	2	B_1	0.3	0.7	0.005	0.9
11	db1	2	3	B_1	0.3	0.7	0.005	0.9
12	db1	2	4	B_1	0.3	0.7	0.005	0.9
13	db1	2	5	B_1	0.3	0.7	0.005	0.9
14	db1	2	6	B_1	0.3	0.7	0.005	0.9
15	db1	2	7	B_1	0.3	0.7	0.005	0.9
16	db1	2	8	B_1	0.3	0.7	0.005	0.9
17	db1	NA	NA	B_1	NA	NA	0.01	0.9
18	db1a	NA	NA	B_1	NA	NA	0.01	0.9
19	db1a	1	2	B_1	0.3	0.7	0.01	0.9
20	db1a	2	2	B_1	0.3	0.7	0.005	0.9
21	db2	NA	NA	B_1	NA	NA	0.01	0.9
22	db2	1	2	B_1	0.3	0.7	0.01	0.9
23	db3	NA	NA	B_1	NA	NA	0.01	0.9
24	db3	1	2	B_1	0.3	0.7	0.01	0.9
25	db4	NA	NA	B_2	NA	NA	0.005	0.9
26	db4	NA	NA	B_1	NA	NA	0.005	0.9
27	db4	1	2	B_1	0.3	0.7	0.01	0.9
28	db4	1	2	B_1	0.7	0.3	0.01	0.9
29	db4	2	2	B_1	0.7	0.3	0.005	0.9
30	db4	2	2	B_1	0.3	0.7	0.005	0.9

We call network A the network that learns augmentations that help train network B.

All experiments are run for 1000 epochs. The test accuracy reported is for the network that had the highest score on the validation set during those 1000 epochs.

To analyze the effectiveness of Smart Augmentation, we performed 30 experiments using 4 datasets with different parameters. A brief overview of the experiments can be seen in Table I. The experiments were conducted with the motivation of answering the following questions:

1) Is there any difference in accuracy between using smart augmentation and not using it? (Is smart augmentation effective?)
2) If smart augmentation is effective, is it effective on a variety of datasets?
3) As the datasets become increasingly unconstrained, does smart augmentation perform better or worse?
4) What is the effect of increasing the number of channels in the smart augmentation method?
5) Can smart augmentation improve accuracy over traditional augmentation?
6) If smart augmentation and traditional augmentation are combined, are the results better or worse than not combining them?
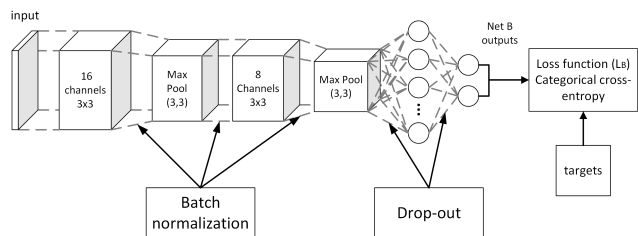
FIGURE 7. Illustration of network B_1.

FIGURE 8. Illustration of network *A*.

7) Does altering the α and β parameters change the results?
8) Does Smart Augmentation increase or decrease over-fitting as measured by train/test loss ratios?
9) If smart augmentation decreases overfitting, can we use it to replace a large complex network with a simpler one without losing accuracy?
10) What is the effect of the number of network A's on the accuracy? Does training separate networks for each class improve the results?

As listed below, we used three neural network architectures with varied parameters and connection mechanisms. In our experiments, these architectures were combined in various ways as specified in table 1.

- Network B_1 is a simple, small Convolutional neural network, trained as a classifier, that takes an image as input, and outputs class labels with a softmax layer. This network is illustrated in figure 7.
- Network B_2 is a unmodified implementation of VGG16 as described in [19]. Network B_2 is a large network that takes an image as input and outputs class labels with a softmax layer.
- Network *A* is a Convolutional neural network that takes one or more images as input and outputs a modified image.

A. SMART AUGMENTATION WITH ONE NETWORK A ON THE GENDER CLASSIFICATION TASK

Experiments 1-8, 19,22, and 24 as seen in table 1 were trained for gender classification using the same technique as illustrated in figure 9. In these experiments, we use smart augmentation to train a network (network B) for gender classification using the specified database.

The first, k images are randomly selected from the same class (male or female) in the dataset. These k samples are merged into k channels of a single sample. The grayscale values of the first image, img_0, are mapped to channel 0 and the grayscale values of the second image im_1 are mapped to channel 1 and so on until we reach the number of channels specified in the experiments table. This new k channel image is fed into the network A. Network A is a fully convolutional neural network (See figure 8) which accepts images as the input and gives the images with the same size at the output in a single channel.
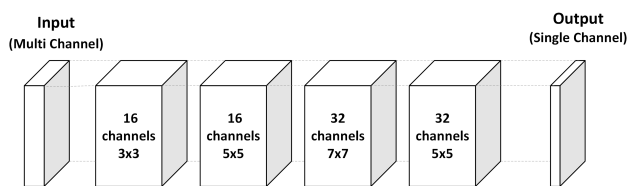
An additional grayscale image is then randomly selected from the same class in the dataset (this image should not be any of those images selected in step 1). The loss function for this network A is calculated as the mean squared error between this randomly selected image and the output of network A. The output of network A, and the target image is then fed into network B as separate inputs. Network B is a typical deep neural network with two convolutional layers followed by batch normalization and max-pooling steps after each convolutional layer. Two fully connected layers are placed at the end of the network. The first of these layers has 1024 units and the second dense layer is made of two units as the output of network B using softmax. Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting. The loss function of network B is calculated as the categorical cross-entropy between the outputs and the targets.

The total loss of the whole model is a linear combination of the loss functions of two networks. This approach is designed to train a network A that generates samples which reduce the error for network B. The validation loss was calculated only for network B, without considering network. This allows us to compare validation loss with and without smart augmentation.

Our models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.01 and momentum 0.9. The lasagne library used to train the network in python.

In these experiments, we varied the number of input channels and datasets used. Specifically, we trained a network B from scratch with 1-8 input channels with a single network A on db1, 2 channels on network A for db2 and 3, and 2 channels on network db1a as shown in the table of experiments.

B. SMART AUGMENTATION WITH TWO NETWORK A's ON THE GENDER CLASSIFICATION TASK

In experiments 9-16 and 20 we evaluate a different implementation of smart augmentation, containing a separate network A for each class. As before, the first k images are randomly selected from the same class (male or female) in the dataset. These k samples are merged into k channels of a single sample. The grayscale values of the first image, img_0, are mapped to channel 0 and the grayscale values of the second image, im_1, are mapped to channel 1, and so on until we reach the number of channels specified in the experiments table just as before. Since we now have two network A's, it is important to separate out the loss functions for each network as illustrated in figure 10.
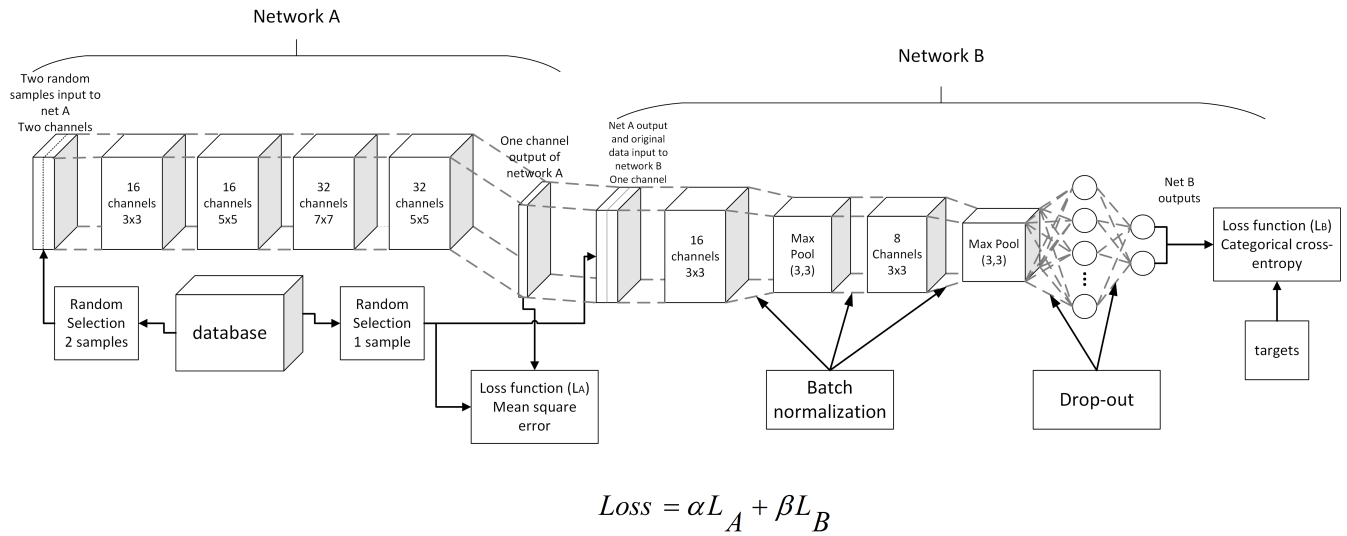
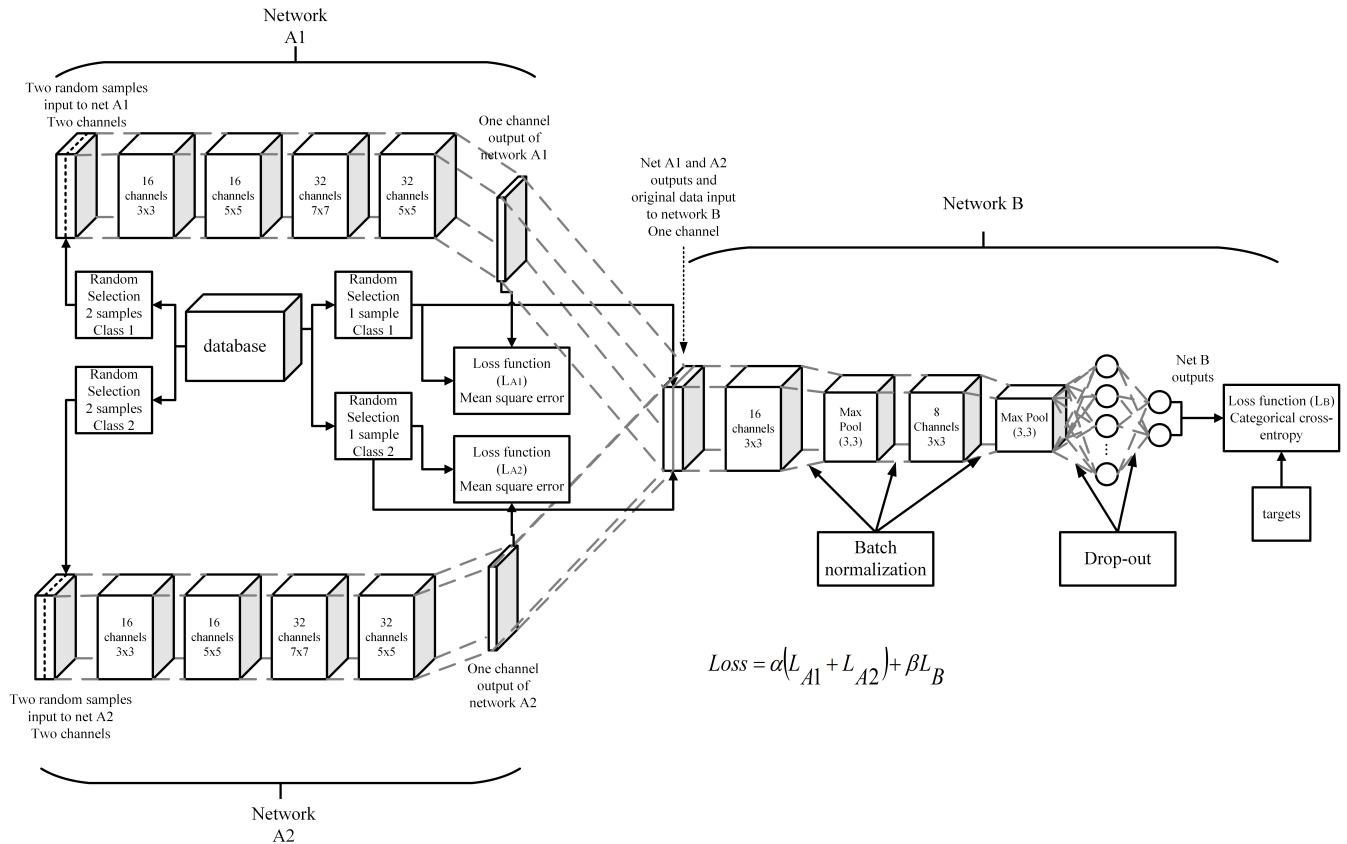**FIGURE 9.** Diagram of simplified implementation of Smart Augmentation showing network A and network B.

$$Loss = \alpha L_A + \beta L_B$$

FIGURE 10. Diagram of our implementation of Smart Augmentation with one network A for each class.

$$Loss = \alpha\left(L_{A1} + L_{A2}\right) + \beta L_B$$

All other loss functions are calculated the same way as before.

One very important difference is the updated learning rate (0.005). While performing initial experiments we noticed that using a learning rate above 0.005 led to the "dying

RELU" problem and stopped effective learning within the first two epochs. This network is also more sensitive to variations in batch size.

The goal of these experiments was to examine how using multiple network As impacts accuracy and overfitting

compared to just using one network A. We also wanted to know if there were any differences when trained on a manually augmented database (experiment 20).
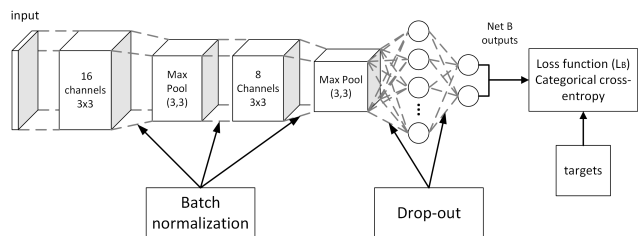

FIGURE 11. Diagram of implementation of network B without Smart Augmentation.

C. TRAINING WITHOUT SMART AUGMENTATION ON THE GENDER CLASSIFICATION TASK

In these experiments, we train a network [network B (see figure 11)] to perform gender classification without applying network A during the training stage. These experiments (23, 21, 18, and 17) are intended to serve as a baseline comparison of what network B can learn without smart augmentation on a specific dataset (db3,db2, db1a, and db1 respectively). In this way, we measure any improvement given by smart augmentation. A full implementation of Network B is shown in figure 7.

This network has the same architecture as the network B presented in the previous experiment except that it does not utilize a network A.

As before, two fully connected layers are placed at the end of the network. The first of these layers has 1024 units, and the second dense layer has two units (one for each class). Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting.

All loss functions (training, validation, and testing loss) were calculated as the categorical cross-entropy between the outputs and the targets.

As before, models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.01 and momentum 0.9. The lasagne library was used to train the network in python.

D. EXPERIMENTS ON THE PLACES DATASET

In the previous experiments in this section, we used 3 different face datasets. In experiments 25 - 30 we examine the suitability of Smart Augmentation with color scenes from around the world from the MIT Places dataset to evaluate our method on data of a completely different topic. We varied the α and β parameter in our global loss function so that we could identify how they influence results. Unlike in previous experiments, we also retained color information.

Experiment 25 utilized a VGG16 trained from scratch as a classifier, chosen because VGG16 models have performed very well on the places dataset in public competitions [16]. The input to network A was 256×256 RGB images and the output was determined by a 2 class softmax classifier.

In experiment 26 we use a network B, identical in all respects to the one used in the previous subsection, except that we use the lower learning rate specified in the experiments table and take in color images about places instead of gender.

These two experiments (25,26) involved simple classifiers to establish a baseline against which other experiments on the same dataset could be evaluated.

In experiments 27-28, k images were randomly selected from the same class (abbey or airport) in the dataset. These k samples are merged into $k \times 3$ channels of a single sample. The values of the first three channels of image img_0 are mapped to channel 0-2, and the first three channels of the second image im_1 are mapped to channels 3-5, and so on, until we reach the number of channels specified in the experiments table multiplied by the number of color channels in the source images. This new $k \times 3$ channel image is used by network A. Network A is a fully convolutional neural network) which accepts images as the input, and outputs just one image.

An additional image is then randomly selected from the same class in the dataset. The loss function for network A is calculated as the mean squared error between the randomly selected image and the output of network A. The output of network A, and the target image is then fed into network B as separate inputs. Network B is a typical deep neural network with two convolutional layers followed by batch normalization and max-pooling steps after each convolutional layer. Two fully connected layers are placed at the end of the network. The first of these layers has 1024 units and the second dense layer is made of two units as the output of network B using softmax. Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting. The loss function of network B is calculated as the categorical cross-entropy between the outputs and the targets.

The total loss of the whole model is a linear combination of the loss functions of two networks. This approach is designed to train a network A that generates samples that reduce the error for network B. The validation loss was calculated only for network B, without considering network A. This allows us to compare validation loss with and without smart augmentation.

Our models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.005 and momentum 0.9. The Lasagne library was used to train the network in python.

In these experiments, we varied the number of input channels and datasets used. Specifically, we trained a network B from scratch with 1-8 input channels on network A on db1, 2 channels on network A for db2 and 3, and 2 channels on network db1a as shown in the table of experiments.

In experiments 29-30, k images are randomly selected from the same class (abbey or airport) in the dataset. These k samples are merged into $k \times 3$ channels of a single sample. The values of the first three channels in image img_0 are mapped to channel 0-2 and the first three channels of the second image im_1 are mapped to channels 3-5 and so

on until we reach the number of channels specified in the experiments table multiplied by the number of color channels in the source images. This new $k \times 3$ channel image is fed into the network A. Network A is a fully convolutional neural network which accepts images as the input and outputs a single color image.

An additional image is then randomly selected from the same class in the dataset. The loss function for each network A is calculated as the mean squared error between the randomly selected image and the output of network A. The output of network A, and the target image is then fed into network B as separate inputs. Network B is a typical deep neural network with two convolutional layers followed by batch normalization and max-pooling steps after each convolutional layer. Two fully connected layers are placed at the end of the network. The first of these layers has 1024 units, and the second dense layer is made of two units as the output of network B using softmax. Each dense layer takes advantage of the drop-out technique in order to avoid over-fitting. The loss function of network B is calculated as the categorical cross-entropy between the outputs and the targets.

The total loss of the whole model is a linear combination of the loss functions of the two networks. This approach is designed to train a network A that generates samples that reduce the error for network B. The validation loss was calculated only for network B, without considering network A. This allows us to compare validation loss with and without smart augmentation.

Our models were trained using Stochastic Gradient Descent with Nesterov Momentum [20], learning rate 0.005 and momentum 0.9. The lasagne library was used to train the network in python.

In these experiments, we varied the number of input channels and datasets used. Specifically, we trained a network B from scratch with 1-8 input channels on network A on db1, 2 channels on network A for db2 and 3, and 2 channels on network db1a as shown in the table of experiments.

VI. RESULTS

The results of experiments 1-30 as shown in Table 1 are listed in tables 2 and 3 and are listed in the same order as in the corresponding experiments table. These results are explained in detail in the subsections below.

A. SMART AUGMENTATION WITH ONE NETWORK A ON THE GENDER CLASSIFICATION TASK

In figure 12, we show the training and validation loss for experiments 1 and 17. As can be observed, the rate of overfitting was greatly reduced when smart augmentation was used compared to when it was not used.

Without smart augmentation, network B had an accuracy of 88.15 for the AR faces dataset; for the rest of this subsection, this result is used as a baseline by which other results on that dataset are evaluated.

One can see how the smart augmentation technique could prevent network B from overfitting in the training stage.

TABLE 2. Results of experiments on face datasets.

Experiments on Face Datasets				
Dataset	#Net As	Input Channels	Augmented	Test Accuracy
AR Faces	1	1	no	0.927746
AR Faces	1	2	no	0.924855
AR Faces	1	3	no	0.950867
AR Faces	1	4	no	0.916185
AR Faces	1	5	no	0.910405
AR Faces	1	6	no	0.933526
AR Faces	1	7	no	0.916185
AR Faces	1	8	no	0.953757
AR Faces	2	1	no	0.869942188
AR Faces	2	2	no	0.956647396
AR Faces	2	3	no	0.942196548
AR Faces	2	4	no	0.942196548
AR Faces	2	5	no	0.907514453
AR Faces	2	6	no	0.933526039
AR Faces	2	7	no	0.916184962
AR Faces	2	8	no	0.924855471
AR Faces	0	NA	no	0.881502867
AR Faces	0	NA	yes	0.890173435
AR Faces	1	2	yes	0.956647396
AR Faces	2	2	yes	0.956647396
Adience	0	NA	no	0.700206399
Adience	1	2	no	0.760577917
FERET	0	NA	no	0.835242271
FERET	1	2	no	0.884581506

TABLE 3. Results of experiments on place dataset.

Experiments on MIT places dataset				
#Net As	Target Network	Test Accuracy	A	B
0	VGG16	98.5	NA	NA
0	Small net B	96.5	NA	NA
1	Small net B	98.75	0.3	0.7
1	Small net B	99%	0.7	0.3
2	Small net B	99%	0.7	0.3
2	Small net B	97.87%	0.3	0.7

The smaller difference between training loss and validation loss caused by the smart augmentation technique shows how this approach helps the network B to learn more general features for this task. Network B also had higher accuracy on the test set when trained with smart augmentation.

In figures 13 and 14 we show examples of the kinds of images network A learned to generate. In these figures, the image on the left side is the blended image of the other two images produced by network A.

We observe an improvement in accuracy from 83.52% to 88.46% from smart augmentation on Feret with 2 inputs and an increase from 70.02% to 76.06% on the adience dataset.

We see that there is no noticeable pattern when we vary the number of inputs for network A. Despite the lack of a pattern, a significant difference was observed with 8 and 3 channels providing the best results at 95.38% and 95.09% respectively. At the lower end, 7, 5, and 4 channels performed the worst, with accuracies of 91.62%, 91.04%, and 91.04%.

Recall that the accuracy without network A was: 88.15% for the AR faces dataset. We suspect that much of the variation in accuracy reported above may be due to chance. Since in this particular experiment, images are chosen randomly there may be times when 2 or more images with very helpful mutual information are present by chance and the opposite is also possible. It is interesting that when 3 and 8 channels were used for network A, the accuracy was over 95%.
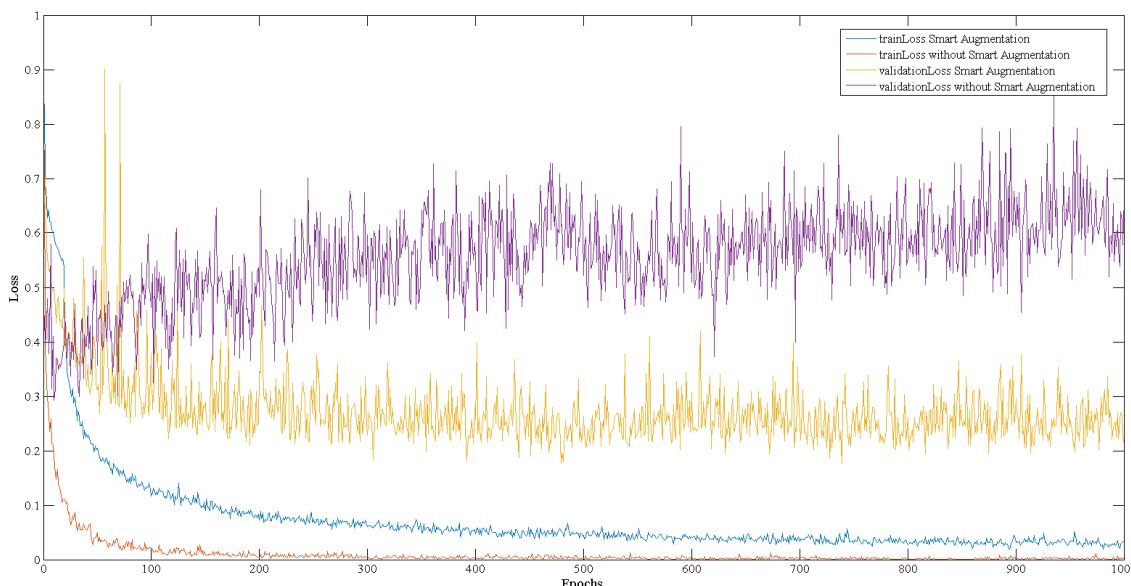
FIGURE 12. Training and validation losses for experiments 1 and 17, showing reductions in overfitting by using Smart Augmentation. The smaller difference between training loss and validation loss caused by the smart augmentation technique shows how this approach helps the network B to learn more general features for this task. To avoid confusion, we remind the reader that the loss for smart augmentation is given by $f(L_A, L_B; \alpha, \beta)$. This means that the loss graphs are a combination of the losses of two networks whereas the losses without smart augmentation are only $f(L_B)$.
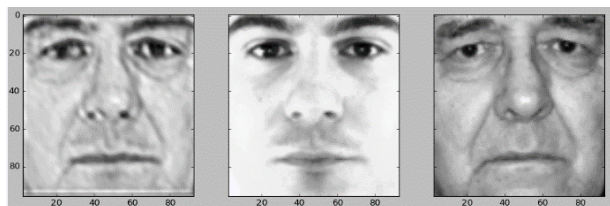

FIGURE 13. The image on the left is a learned combination of the two images on the right as produced by network A.

FIGURE 14. The image on the left is a learned combination of the two images on the right as produced by network A.

B. SMART AUGMENTATION AND TRADITIONAL AUGMENTATION

We note that traditional augmentation improved the accuracy from 88.15% to 89.08% without smart augmentation on the gender classification task. When we add smart augmentation we realize an improvement in accuracy to 95.66%.

The accuracy of the same experiment when we used 2 networks A's was also 95.66% which seems to indicate that both configurations may have found the same optima when smart augmentation was combined with traditional augmentation.

This demonstrates that smart augmentation can be used with traditional augmentation to further improve accuracy. In all cases examined so far, Smart Augmentation performed better than traditional augmentation. However, since there are no practical limits on the types of traditional augmentation that can be performed, there is no way to guarantee that manual augmentation could not find a better augmentation strategy. This is not a major concern since we do not claim that smart augmentation should replace traditional augmentation. We only claim that smart augmentation can help with regularization.

C. SMART AUGMENTATION WITH TWO NETWORK A's ON THE GENDER CLASSIFICATION TASK

In this subsection, we discuss the results of our two network architecture when trained on the gender classification set.

These experiments show that approaches which use a distinct network A for each class, tend to slightly outperform networks with just 1 network A. This seems to provide support for our initial idea that one network A should be used for each class so that class-specific augmentations could be more efficiently learned. If the networks with just 1 and 0 input channels are excluded, we see an average increase in accuracy from 92.94% to 93.19% when smart augmentation is used, with the median accuracy going from 92.49% to 93.35%.

There is only one experiment where smart augmentation performed worse than not using smart augmentation. This can be seen in the 9th row of table II where we use only one channel which caused the accuracy to dip to 86.99%, contrasted with 88.15% when no smart augmentation is used. This is expected because when only one channel is used, mutual

information can not be effectively utilized. This experiment shows the importance of always using at least 2 channels.

D. EXPERIMENTS ON THE PLACES DATASET

As with previously discussed results, when the places dataset is used, networks with multiple network A's performed slightly better. We also notice that when α is higher than β an increase in accuracy is realized.

The most significant results of this set of experiments is the comparison between smart augmentation, VGG 16, and network B trained alone. Note that a small network B trained alone (no Smart Augmentation) had an accuracy of 96.5% compared to VGG 16 (no Smart Augmentation) at 98.5%. When the same small network B was trained with smart augmentation we see accuracies ranging from 98.75% to 99% which indicates that smart augmentation, in some cases, can allow a much smaller network to replace a larger network.

VII. DISCUSSION AND CONCLUSION

Smart Augmentation has shown the potential to increase accuracy by demonstrably significant measures on all datasets tested. In addition, it has shown potential to achieve similar or improved performance levels with significantly smaller network sizes in a number of tested cases.

In this paper, we discussed a new regularization approach, called "Smart Augmentation" to automatically learn suitable augmentations during the process of training a deep neural network. We focus on learning augmentations that take advantage of the mutual information within a class. The proposed solution was tested on progressively more difficult datasets starting with a highly constrained face database and ending with a highly complex and unconstrained database of places. The various experiments presented in this work demonstrate that our method is appropriate for a wide range of tasks and demonstrates that it is not biased to any particular type of image data.

As a primary conclusion, these experiments demonstrate that the augmentation process can be automated, specifically in nontrivial cases where two or more samples of a certain class are merged in nonlinear ways resulting in improved generalization of a target network. The results indicate that a deep neural network can be used to learn the augmentation task in this way at the same time the task is being learned. We have demonstrated that smart augmentation can be used to reduce overfitting during the training process and reduce the error during testing.

It is worthwhile to summarize a number of additional observations and conclusions from the various experiments documented in this research.

Firstly, no linear correlation between the number of samples mixed by network A and accuracy was found so long as at least 2 samples are used.

Secondly, it was shown that Smart Augmentation is effective at reducing error and decreasing overfitting and that this is true regardless of how unconstrained the database is.

Thirdly, these experiments demonstrated that better accuracy could be achieved with smart augmentation than with traditional augmentation alone. It was found that altering the α and β parameters of the loss function slightly impacts results but more experiments are needed to identify if optimal parameters can be found.

Finally, it was found that Smart Augmentation on a small network achieved better results than those obtained by a much larger network (VGG 16). This will help enable more practical implementations of CNN networks for use in embedded systems and consumer devices where the large size of these networks can limit their usefulness.

Future work may include expanding Smart Augmentation to learn more sophisticated augmentation strategies and performing experiments on larger datasets with larger numbers of data classes. A statistical study to identify the number of channels that give the highest probability of obtaining optimal results could also be useful.

REFERENCES

[1] S. Bazrafkan, T. Nedelcu, P. Filipczuk, and P. Corcoran, "Deep learning for facial expression recognition: A step closer to a smartphone that knows your moods," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2017, pp. 217–220.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[3] S. Ioffe and C. Szegedy. (2015). "Batch normalization: Accelerating deep network training by reducing internal covariate shift." [Online]. Available: https://arxiv.org/abs/1502.03167

[4] P. Y. Simard, D. Steinkraus, and J. C. Platt, "Best practices for convolutional neural networks applied to visual document analysis," in *Proc. ICDAR*, vol. 3. Aug. 2003, pp. 958–962.

[5] F. Chollet. (2015). *Keras*. [Online]. Available: https://github.com/fchollet/keras

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

[7] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. (2014). "Return of the devil in the details: Delving deep into convolutional nets." [Online]. Available: http://arxiv.org/abs/1405.3531

[8] X. Bouthillier, K. Konda, P. Vincent, and R. Memisevic. (2015). "Dropout as data augmentation." [Online]. Available: https://arxiv.org/abs/1506.08700

[9] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. (2014). "Synthetic data and artificial neural networks for natural scene text recognition." [Online]. Available: http://arxiv.org/abs/1406.2227

[10] I. J. Goodfellow, J. Shlens, and C. Szegedy. (2014). "Explaining and harnessing adversarial examples." [Online]. Available: https://arxiv.org/abs/1412.6572

[11] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[12] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. (2016). "Learning from simulated and unsupervised images through adversarial training." [Online]. Available: https://arxiv.org/abs/1612.07828

[13] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Trans. Knowl. Data Eng.*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[14] Theano Development Team. (May 2016). "Theano: A python framework for fast computation of mathematical expressions." [Online]. Available: http://arxiv.org/abs/1605.02688

[15] J. Lemley, S. Abdul-Wahid, D. Banik, and R. Andonie, "Comparison of recent machine learning techniques for gender recognition from facial images," in *Proc. 27th Modern Artif. Intell. Cognit. Sci. Conf.*, 2016, pp. 97–102.

[16] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 487–495.

[17] A. Martinez and R. Benavente, "The AR face database," CVC, Tech. Rep. #24, 1998. [Online]. Available: http://www2.ece.ohio-state.edu/~aleix/ARdatabase.html

[18] P. J. Phillips, H. Moon, S. A. Rizvi, and P. J. Rauss, "The FERET evaluation methodology for face-recognition algorithms," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 10, pp. 1090–1104, Oct. 2000.

[19] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[20] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proc. ICML*, Feb. 2013, vol. 28. no. 3, pp. 1139–1147.

JOSEPH LEMLEY (S'17) received the B.S. degree in computer science and the master's degree in computational science from Central Washington University in 2006 and 2016, respectively. He is currently pursuing the Ph.D. degree with the National University of Ireland Galway. He was with the industry and co-founded a start-up. His field of work is machine learning using deep neural networks for tasks related to computer vision. His Ph.D. was funded by FotoNation, Ltd., under the IRCSET Employment Ph.D. Program.

SHABAB BAZRAFKAN (S'16) received the B.Sc. degree in electrical engineering from Urmia University, Urmia, Iran, in 2011, and the M.Sc. degree in telecommunication engineering, image processing branch from the Shiraz University of Technology in 2013. He is currently pursuing the Ph.D. degree with the National University of Ireland Galway. He is currently with FotoNation, Ltd. His field of working is deep neural networks and neural network design.

PETER CORCORAN (M'95–F'10) was the former Vice-Dean of Research and Graduate Studies (seven years tenure) with the College of Engineering and Informatics, National University of Ireland Galway. He has been a University Professor for 28 years. He is the co-founder of several start-up companies including FotoNation, industry consultant and expert witness. He has over 500 technical publications and patents, over 80 peer reviewed papers and articles, over 100 international conference papers. He has co-invented over 300 granted U.S. patents. He is a member of the IEEE Consumer Electronics Society for over 20 years. He is the Editor-in-Chief and the Founding Editor of the *IEEE Consumer Electronics Magazine*.

• • •