DAM: An Adversarial-based Approach to Enhance Robustness Using Mix-up Methods

***Abstract*—To increase robustness and reduce overfitting, usually, convolutional neural networks are trained leveraging augmented images generated by label-preserving transformations of original training image data. Although these transformations do increase performance, recent studies have shown that non-label preserving data augmentation can be remarkably efficient and further increase performance. In this study we work to further the baseline classifier, we explore the mix-up between images in three variants: linear mix-up, horizontal concatenations, and vertical concatenations. We have also accommodated adversarial training for each of the three mix-up variants to protect the images from misclassifications and adversarial attacks. This model outperforms the baseline model by 68.8% for noisy images. Additionally, two types of filters are being used to enhance the model's classification of the target object: Attention filter and Environmental filter. This is the first of its kind as an all-round approach to providing the best image classifications using non-label preserving data augmentation.**

I. Introduction

Neural networks have been proven time and again to be extremely useful tools in the domain of computer vision [1], [2]. Specifically, neural networks can be trained to classify images based on objects present in the images to a very high degree of accuracy [3]. However, there are still concerns when it comes to training these neural networks as image classifiers. Overfitting and the network's performance on data outside the training set is always a concern when training any neural network, and image classifiers are no different [4]. Another major concern is the network's robustness in handling adversarial attacks [5]. It has shown that the performance of image classifiers can be severely diminished by relatively simply adversarial attacks [6]. In this work, we seek to formulate a general process during training that can improve a network's testing accuracy on data it was not trained on, as well as defending against adversarial attacks.

In our work to formulate a process to improve the quality of image classifying neural networks we have implemented a pipeline through which the training data is sent. Our method is built on three high level steps: applying filters to images, augmenting images and defensive adversarial training. In the first step, we take training images and have the ability to apply either Attention filters, environmental filters, both filters, or neither. The details of these filters and how they are utilized are covered in Section II-A. It is our hope that by applying these filters the network can learn what the important object is in the image and how to pick out the object in varying environmental situations. The next stage takes images and applies augmentations. The augmentations that were specifically used in our pipeline are "mix-up" methods which work by taking multiple original images from the training back and combining them in various linear or non-linear ways. Mix-up methods have been shown to be more effective than traditional image data augmentations techniques such as cropping, rotations and flips, [7] and as such we attempted to integrate them with our methods, the details of this step can be found in Section II-B. Finally, we use defensive adversarial training on the mixed-up images in order to improve the robustness of our trained image classifier. Since we are passing mixed-images to the defensive adversarial training, we must apply this defense in a way that is unique to the mix-up method that was applied to the data, we call this method Defensive Adversarial Mix-up (DAM) and the details of this can be found in Section II-C.

Because our pipeline has so many components it was necessary to have many different test cases in evaluating what worked and what did not work to improve the network's performance and robustness. Section III gives an overview of the results of our method with numerous different combinations of parts of our pipeline turned on and off. In general, we tested the performance of our models on both clean training data, and training data that has some adversarial noise applied to and used baseline vanilla classifiers as a performance benchmark. In this way we can see how our methods do in our goal of improving performance on training sets and defending against adversarial attacks.

A. Novelties

In summary, This study contains the following novelties:

1) A novel Defensive Adversarial Mix-up (DAM) method to improve the robustness of the system against noisy data is proposed.
2) A novel approach for training models using DAM method using variable ϵ is proposed to alleviate the declining accuracy for noisy data.
3) Multiple mixup functions have been tested to investigate their effect on the overall effectiveness of DAM method.
4) An attention-based filter for input images has been developed to be used for training the model to explore the possibility of increasing accuracy and robustness of the models.
5) An environmental filter for input images has been developed to be used for increasing the robustness of the system for environmental changes in the input images.

B. Literature Background

The conception of this project was based upon the work done by Cecilia Summer and Michael J. Dinneen in "Improved

Mixed-Example Data Augmentation" [7]. In this work the authors test the performance of mixed target image data augmentation when training an image classifying neural network. The idea behind these methods is to take images belonging to various classes and combine them into one image. This new image is then treated as a new data point for the purposes of training, and the label is taken to be a mix of the original classes. Specifically, the authors study mixing images in both linear and non-linear ways. In the work, they survey 14 such linear and non-linear image mixing techniques and show that most of them result in improved neural network performance over more classical data augmentations. This is an interesting result because while traditional augmentations to image data (crops, rotations and flips) are very intuitive to humans in how they can help a neural network generalize to data outside the training set, these mixing methods are not as intuitive. Originally, the mix-up method was first introduced in [17], in which just the Linear Mix-up method was applied; however, [7] studied multiple linear and non-linear methods.

These results inspired us to take a further look at how image data can be augmented/manipulated in the process of training an image classifier, and how performance can be improved by doing so.

II. Experiment Design

Cecelia Summers and Micheal J. Dinneen have explored a variety of linear and non-linear methods of image mixing across classes in their baseline model (RESNET+Mix-Up) [7]. By amalgamating images from different classes and providing a mixed target value for each of the images, the image classifying accuracy of the CNN model was significantly improved. In this project, we study the potential for improving the existing approach and formulated components that can be implemented to improve the image classifier's efficiency. In this section, we discuss the architecture of the Defensive Adversarial Mix-Up (DAM) model and the individual components that make this technique novel.

Image filters, image mix-up, and defensive adversarial mix-up are the three main components of the DAM strategy. Figure 1 depicts the DAM technique's modular architecture. Image filtering is the first component of the model and is divided into two sub-modules: attention filtering and environmental filtering. The aim of attention filters is to highlight the important sections of the input image, such as the target object, while fading out the surrounding regions. The attention approach is explained in detail in section II-A1. Environmental filters, in contrast to attention filters, aim to augment several surrounding scenarios for the target object, and it is performed in three variants: occlusion scenarios, road conditions, and weather conditions. The goal of filtering is to aid the neural network model in determining the object's importance in relation to its surroundings; a clear explanation of each variant can be found in section II-A2.

In the image Mix-up portion of our model, we perform four different types of mix-ups after acquiring the filtered images:

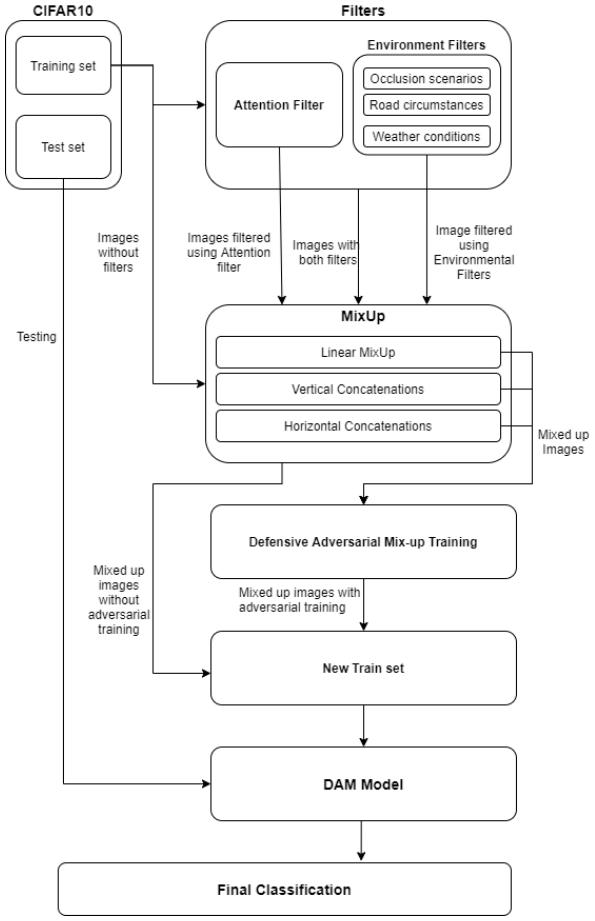

Fig. 1: Defensive Adversarial Mix-Up (DAM) Architecture

Linear Mix-Up, Vertical Concatenations and Horizontal Concatenations. In section II-A, the algorithms for performing the mix-ups are discussed, and in section II-C, the strategies used for defensive adversarial training for each of the mix-ups are detailed.

In this project we utilize these three modules, as well as the sub-modules inside them, in different combinations and study the performance for image classification.

A. Image Filters

1) Attention Filtering: As the first implemented filter in our architecture, we decided to use attention. Attention is the effort to mimic the selective concentration function in human brain in the context of neural networks. For images the concept has been proposed as feature localization technique [12] and the goal was to determine the location of the image where the object resides. The applications of attention have revolutionized the industry by the introduction of self-attention, as the core of the transformer-based models as one of the most popular architectures used in many applications. Attention is widely used in various applications such as NLP [15], machine vision [14], and speech recognition [13]. Attention aims to learn the relative importance of different parts of input and only focus on the important fragments.

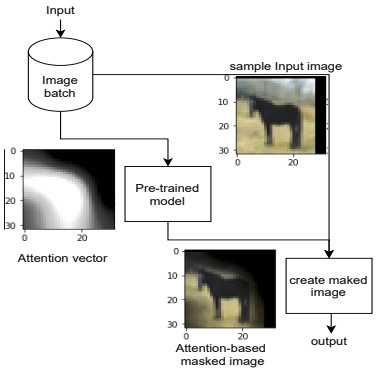

Fig. 2: A simplified architecture of attention filter and its effect on a sample image

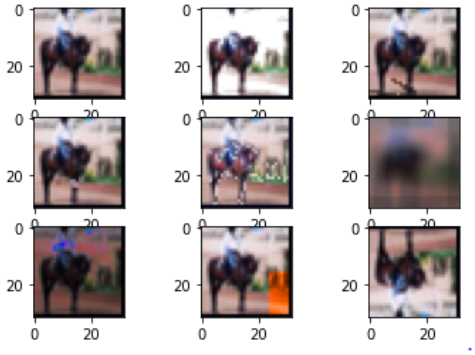

Fig. 3: A random sample of augmentation techniques performed on the training images. The images have been subject to augmentations in the following order from top row to the bottom row (left to right): original image, presence of sunlight (brighten), shadowed image, presence of gravel, addition of snow, addition of fog, night time(darken), autumn season and a random image flip(inverted)

The primary concept behind the attention is to let find which part of the inputs have more information for our objective and highlight the crucial information. In this work, our idea is to use one attention mechanism to filter out the surrounding area of the objects in the images of our dataset. As a result of this omission of unrelated objects in the images (such as the background), the model performs better in classifying the images which in turn improves the accuracy as well as the robustness of our model.

To implement our original idea, we added the attention filter. We used the code from [16] as an implementation of simple CNN with Class Activation Maps (CAM). CAM is a feature localization approach which was proposed in 2015 by Wan et al. In this work, the authors proposed an algorithm to generate Class Activation Maps (CAM) using Global Average Pooling (GAP) in CNNs. To this end, for each unit at the last CNN, they outputted the spatial average of the feature map using the global average pooling. Then, they calculated the average sum of these values to obtain CAMs. This procedure can be formulated as follows:

$$S_c = \sum_{x,y} \sum_{k} w_k^c f_k(x.y) \tag{1}$$

Where w_c^k is the weights and $f_k(x, y)$ is the output of last layer of CNN at spatial location (x,y). We train a model using their approach and then use the trained model on the CIFAR-10 dataset as the core of our attention filter. In this filter, we feed input each image into this pre-trained model, retrieve the attention vector of the image which its elements are between zero and one and using this attention vector, we mask our image. Figure 2 depicts the structure of this filter as well as a sample output of this component on a sample CIFAR-10 image.

2) Environmental Filtering: The CIFAR-10 dataset contains 60000 images divided into ten groups, including automobiles, birds, cats, and dogs, to name a few. To minimize the overfitting of the model to the training collection, the training images are usually flipped, rotated, scaled, or cropped. In the case of the CIFAR-10 dataset, the images were primarily depicted with a surrounding environment, with numerous other external parameters present alongside the target object. The environment in which an object appears in the train set can differ from the environment in the test set; this lack of knowledge of environmental parameters can result in misclassifications [8]. As a result, an effort is made to improve the CNN model by training it with images that have had different degrees of occlusion, weather, and road condition filters applied to them.

A general set of steps is followed for each form of augmentation filter, as outlined in the work done by Manu Goyal et. al [9]. To begin, the region of interest (ROI) is defined, here being the classification object. Then, the HSB (hue, saturation, and brightness) and RGB (red, blue, green) parameters are updated on the ROI or around the ROI, depending on the augmentation requirement. For example, to apply an 'autumn season' filter, we first mark the object and then the surrounding areas (outside the ROI) are scaled up in the R channel of the RGB values. Our project uses random environmental filter techniques to improve the training set. 3 depicts an example of such augmentations.

To conduct the image augmentations, several environmental parameters were taken into account [10]. Fog and rain, for example, have a significant impact on the image's visibility. The addition of gravel, the existence of manholes, the speed of the object, the shadows cast by the object, and other techniques are all examples of road augmentation techniques. There are also other factors to consider when it comes to weather, such as different seasons like fall, spring, and winter (snow), as well as different brightness levels, such as full bright, partial sunflare, and night time. These are some of the augmentation scenarios that were considered for this research project's environmental augmentation filters. Each parameter's sensitivity can be fine-tuned manually, but this project also includes automated tuning. The method of implementation and area of coverage in the image are updated based on the relevant augmentation criteria. It's worth noting that certain augmentations, such as snow, must be present on both the target object and the surrounding area. These parameters can be modified as the augmentation

coefficients change. If the coefficients are relatively small, the augmentation is only performed on the surrounding pixels, leaving the pixels containing the target object undisturbed.

B. Image Mix-up

In this section we give a detailed overview of the image mix-up techniques that we applied in our project. As mentioned above, various techniques belonging to the image mixing family were surveyed by Summers and Dinneen and were found to improve the performance of image classifying neural networks [7]. All of these mixing techniques involved taking two images from the training set and producing a new image based on some weighted combination of the original two. The mixed images are then used as a training instance for the network, while the training target is a weighted combination of the two original labels (where the weights are the same as those used in the image mixing). In this work, we utilize some of these mixture techniques while also extending them to novel forms by mixing a general number of training images together, rather than just two.

1) Linear Mix-up: In this technique a new image is produced by the element-wise summation of a given number of original images. These images have been randomly selected and have all been individually scaled by some weight factor in the range [0,1]. Additionally, all of these weights sum to 1.0, such that we can think of the new image as a weighed average of the originals. This augmentation can be expressed as:

$$\tilde{\boldsymbol{x}} = \sum_{i=1}^{n} \lambda_i \boldsymbol{x}_i \tag{2}$$

Where $\tilde{\boldsymbol{x}}$ is the mixed image, the x_i's are the randomly selected images from the training data, n is the number of images to be mixed and the λ_i's are the weights sampled from an n-dimensional Dirichlet distribution.

2) Vertical Concatenations: In this augmentation images are mixed by slicing out a random number of rows from a given number of original images and then concatenating these rows. The number of rows taken from each individual original image is given by weights assigned to each image. Again, like the linear mix-up technique discussed above, these weights are in the range [0,1] and all must sum to 1.0. These requirements are necessary to ensure the new mixed image has consistent dimensions with the images in the original data set. This augmentation can be expressed as:

$$\tilde{\boldsymbol{x}} = \sum_{i=1}^{n} \begin{cases} \boldsymbol{x}_i(r,c), & \text{when } r_{i-1} \leq r \leq r_{i-1} + \lambda_i H \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Where $\tilde{\boldsymbol{x}}$ is the mixed image, the $\boldsymbol{x}_i$'s are the randomly selected images from the training data, n is the number of images to be mixed, r and c are the rows and columns of the images respectively, r_i is the row where the previous concatenation ended (r_0 = 0), H is the height of the images, and the λ_i's are the weights sampled from an n-dimensional Dirichlet distribution

3) Horizontal Concatenations: This augmentation technique is very similar to the vertical concatenations discussed above. The difference for this technique is that columns from the original images are concatenated to form the mixed image, rather than rows:

$$\tilde{\boldsymbol{x}} = \sum_{i=1}^{n} \begin{cases} \boldsymbol{x}_i(r,c), & \text{when } c_{i-1} \leq c \leq c_{i-1} + \lambda_i W \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Where we have all the same variables as in vertical concatenations except W, which is the width of the image, and c_0, which is equal to 0.

Since these augmentations are mixing images of various classes it is also required to use an updated loss function when training the network. This new loss function needs to take into account the weighted combination of the labels of the mixed images. To do this, we again generalize the loss used by Summers and Dinneen from two mixed labels to any given number:

$$L_{total} = \sum_{i=1}^{n} \lambda_i L(t_i, y) \tag{5}$$

Where L_{total} is the total loss, t_i is the target for the i'th mixed image, λ_i is the weight the i'th image was assigned, y is the network output and $L(t_i, y)$ is the loss for the i'th target with the network output.

Examples of all four image mixing augmentations can be seen in Figure 4. In this demonstration linear mix-up and both concatenations mixed five images from the original batch, the first five mixed images are shown along with the original image.

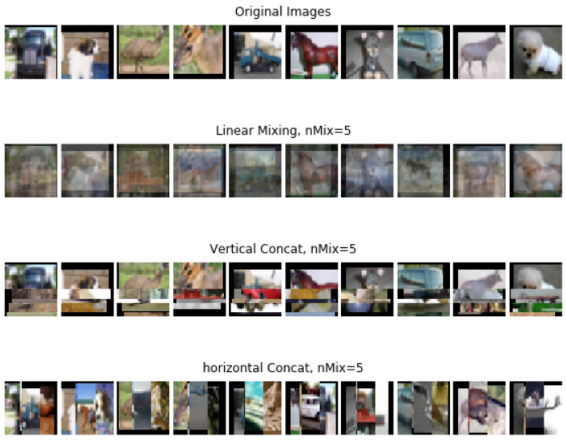

Fig. 4: Demonstration of mixing augmentations. Original images (top row), linear mix-up (second row), vertical concatenations (third row) and horizontal concatenations (fourth row) all mixed five original batched images

C. Defensive Adversarial Mix-up

For the final step in our process we have introduced a new method of adversarial training for defending against

adversarial attacks. We call this method DAM (Defensive Adversarial Mix-up). DAM is a training method that tries to improve the robustness of the model while not sacrificing performance on clean data (i.e., data without adversarial noise). We have studied our model in detail in section III and have shown that this method can dramatically enhance the model's robustness compared to the base Linear Mix-up model and vanilla adversarial training methods. The novelty of the DAM method comes from the fact that it is unique to the mix-up that has been applied to the data and that it can be used with any type of mix-up function.

Our method uses FGSM method [6] for generating adversarial noise during the training. Equations 6 describe this process.

$$\boldsymbol{\eta} = \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right) \tag{6}$$

$\nabla_{\boldsymbol{x}} J$ is gradient of the cost function J with respect to inputs $\boldsymbol{x}$. y is the target for input $\boldsymbol{x}$, and $\boldsymbol{\theta}$ is the model's parameters. $\boldsymbol{\eta}$ is the generated noise with respect to input $\boldsymbol{x}$. For generating an adversarial input, this noise can be added to the clean input. Equation 6 demonstrates that the L^{∞} norm of the difference between the original image and the adversarially perturbed image is equal to ϵ. For obtaining the adversarial input we simply add $\boldsymbol{\eta}$ to the clean input.

The way DAM varies from previous methods is that instead of calculating noise for a base input image x, we must now calculate an adversarial noise with respect to the mix-up image, x_{mixed}. To do this we follow a general set of steps for any mix-up method used in our pipeline. We first calculate the gradient of the loss function with respect to all input images that are being mixed together, as in equation 6. After this we have n adversarial noise with respect to n images, where n is the number of images being mixed. We then combine these tensors with the exact same method with which we combined the base images to produced the mixed image itself. That is, for linear mix up we combine the adversarial noise by equation 2, for vertical concatenations we combine them with equation 3 and for horizontal concatenations we combine them with equation 4. Where in all of these equations x_i is replaced by $\epsilon \operatorname{sign}(\nabla_{\boldsymbol{x_i}} J(\boldsymbol{\theta}, \boldsymbol{x_i}, y))$. Once we have the properly mixed adversarial noise we add it to the mixed image before feeding to the the model. Then the loss is calculated by Equation 5 as before.

III. Evaluation

For the base model used the Resnet18 model [11]. All experiments are tested on the CIFAR-10 dataset, which contains ten different classes.

A. DAM Effect on the Performance

To measure the robustness of the model we trained using the DAM method, we have tested the model on perturbed test data with a specific adversarial noise added to each image. As described in Section II-C, we need to calculate the adversarial noise of input images in the DAM method with the parameter ϵ during the training phase of DAM. We call this parameter coefficient training epsilon. Moreover, another epsilon is needed for generating adversarial inputs while testing the robustness of the model. Although these two variables seems similar, they are different since the former is used to train the system and the latter is used to test the robustness of the trained model. In table I accuracy of different models is compared for different values of epsilon in testing and training phases. In this experiment, the DAM model is using the Linear mixup function mixing two images. The Adversarially Trained Baseline model has the same architecture as the base model (ResNet18) but it is trained using the loss function in equation 7 which is introduced in [6]. This approach of adversarial defense will make the model more robust against the adversarial attacks. The training ϵ is set to 0.35 and the α is set to 0.5. Notice that this vanilla adversarially trained model does not have any mixup method in its training. The Linear Mix-up model is the traditional linear mix-up model used in [7]. The Vanilla Classifier is just the base ResNet18 model trained on CIFAR-10 without any mix-up or adversarial defense.

$$\begin{aligned}\tilde{J}(\boldsymbol{\theta}, \boldsymbol{x}, y) = \alpha J(\boldsymbol{\theta}, \boldsymbol{x}, y) \\ + (1-\alpha) J\left(\boldsymbol{\theta}, \boldsymbol{x} + \epsilon \operatorname{sign}\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y)\right)\right)\end{aligned} \tag{7}$$

TABLE I: Testing the DAM method on adversarial inputs generated using the FGSM adversarial noise generation method (Testing ϵ = 0.35) on CIFAR-10

Model	***Acc. on Clean input (Testing ϵ = 0)***	***Acc. on adversarial input (Testing ϵ = 0.35)***
DAM (Training ϵ = 0.35)	88.3%	81.72%
Vanilla Adversarially Trained Baseline Model (Training ϵ = 0.35)	92.77%	27.56%
Vanilla Classifer with Linear Mixup	95.26%	19.25%
Vanilla Classifier	94.66%	12.64%

Table I illustrates that the DAM method increases the robustness of the model dramatically while preserving the performance of the model on the clean data (88.3%). In comparison with the vanilla adversarially trained model, the DAM model enhances the accuracy on adversarial inputs by 54%.

After analyzing the DAM model on multiple test data with different adversarial noises (i.e., different testing ϵ values), we found that the DAM model does not have consistent behavior. In Essence, we expected to see a decrease in the accuracy as the extend of noise increases (i.e., higher testing ϵ). However, the robustness of the model did not change as we anticipated. In some cases, by increasing the value of testing ϵ, the robustness of the model increased. In order to get a better understanding of our observations, we tested the behavior of the DAM and other models with different amounts of noise (i.e., different values of ϵ) added to the test dataset. The results are shown in Figure 5.

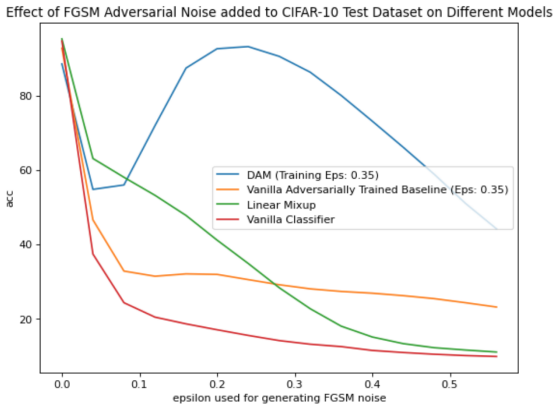

Fig. 5: Behaviour of tested models on different extents of noise

As can be seen in Figure 5, the accuracy of the DAM model (trained using ϵ = 0.35) decreases at first (by increasing the amount of noise), but then it increases.

B. Various Training ϵ for DAM

Figure 5 illustrates that the robustness of the DAM model is not decreasing gradually as the testing ϵ increases. We tried to use the DAM method to train models more resistant and consistent to changes in amount of adversarial noise. Training DAM models with different training ϵ better revealed the behavior of this type of defensive method.

TABLE II: Testing various DAM models on adversarial inputs generated using the FGSM adversarial noise generation method (Testing ϵ = 0.25) on CIFAR-10

Model	***Acc on Clean input (Testing ϵ = 0)***	***Acc on Adversarial input (Testing ϵ = 0.25)***
DAM (Training ϵ = 0.35)	88.3%	81.72%
DAM (Training ϵ = 0.05)	92.53%	45.11%
DAM (Training ϵ = 0.25)	89.41%	88.09%
DAM (Training ϵ = 0.35)	88.53%	92.87%
DAM (Training ϵ = 0.45)	84.32%	95.39%
DAM (ϵ sampled form Normal Distribution (μ = 0.35, σ = 0.08))	90.38%	97.10%
DAM (ϵ sampled form Beta Distribution (α = 1., β = 1.))	90.32%	67.54%
Vanilla Adversarially Trained Baseline Model (Training ϵ = 0.35)	92.77%	30.11%

Table II shows multiple models that are trained using the DAM method with different training ϵ. All these DAM models used linear mix-up as their mixing function with two images mixed. Multiple training ϵ were tested during the experiments. Our results indicated that not having a constant training ϵ might improve the resistance and consistency of the model. To test this hypothesis, we used Normal Distribution and Beta Distribution for sampling the training ϵ during the training. To elaborate more, during the training in each batch, an ϵ is randomly sampled from Normal or Beta Distribution. Figure 6 illustrates how sampling the training ϵ from Normal Distribution resulted in improved performance for our model in terms of accuracy, robustness, and resistance to variation of testing ϵ. As Table II depicts, sampling from Normal Distribution has the second-best performance on clean inputs among various DAM models while it has the best performance on adversarial inputs with an accuracy of 97.10%. One interesting point is that the DAM model with sampling ϵ from Normal Distribution achieves its highest accuracy on some adversarial inputs instead of clean inputs, which is really useful as in many real-world applications the data is noisy to some extent.

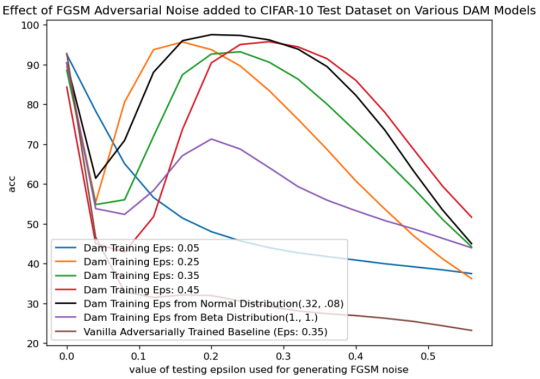

Fig. 6: Behaviour of various DAM models on different extents of noise

C. Various Mixing Functions with DAM

Linear mix-up was used as the mixing function in all experiments explained above and we limited the mixture to just two images. In this experiment, our goal was to investigate the effect of using various mixing functions, using 2 and 3 images. We also wanted to know how using DAM method on top of different mix-up functions may affect our models. To this end, three mixing functions are examined, namely, Linear Mix-up, Vertical Concatenations, and Horizontal Concatenations. Each of these mixing methods can be used with or without the DAM method. Moreover, we varied the number of mixing images to be either 2 or 3. Table III demonstrates the performance of these multiple combinations of mixing functions, number of mixed images, and training with or without DAM. This table shows that among all mixing functions used with DAM, the Linear Mix-up method using two images for mixing has the best accuracy on clean and adversarial inputs with accuracy 90.38 and 97.10, respectively. As mentioned previously, the amount of noise added is very important for every conclusion, so a better illustration for the behaviour of these models is illustrated in Figure 7. It is worth mentioning that only models trained with the DAM method are compared in Figure 7.

Figure 7 illustrates that using the Linear Mixing method with the DAM method has the best robustness and consistency as the amount of noise (i.e., the value of ϵ) increases. In all robust models trained with the DAM method, by increasing the testing ϵ, the accuracy of the model drops and then it starts to rise until a specific point. This area of decrease is

TABLE III: Testing various Mixing Functions (mixing 2 or 3 images) trained with/without DAM method on adversarial inputs generated using the FGSM adversarial noise generation method (Testing ϵ = 0.25) on CIFAR-10. In DAM method for all models the training ϵ has been sampled from Normal Distribution (μ = 0.35, σ = 0.08)

Model Mixup Function	**# Mixed Images**	**DAM (yes/no)**	***Acc. on Clean Input (Testing ϵ = 0)***	***Acc. on Adversarial Input (Testing ϵ = 0.25)***
Linear	2	✗	**95.26%**	33.33%
Linear	2	✓	90.38%	**97.10%**
Linear	3	✗	94.60%	31.65%
Linear	3	✓	90.37%	89.67%
Horizontal	2	✗	94.53%	26.05%
Horizontal	2	✓	88.31%	67.13%
Horizontal	3	✗	94.17%	25.23%
Horizontal	3	✓	86.25%	41.70
Vertical	2	✗	95.24%	27.58%
Vertical	2	✓	86.94%	63.95%
Vertical	3	✗	95.02%	24.52%
Vertical	3	✓	85.13%	57.83%

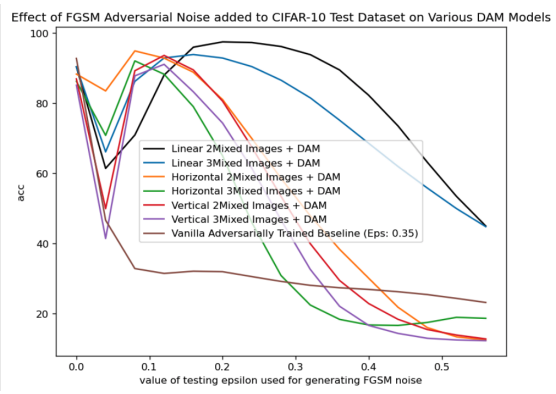

Fig. 7: Behaviour of different DAM models trained with different filters with or without mixing functions on different extents of noise. In all DAM models the training ϵ has been sampled from Normal Distribution (μ = 0.35, σ = 0.08)

approximately when the testing ϵ is between zero and 0.1. The Horizontal Concatenation mixing function showed the best robustness in this area compared to all models; however, when the testing ϵ increases more, its performance diminishes considerably compared to Linear Mixing models.

D. Various Filters with DAM

For our next experiment, we tested two different filters that we introduced earlier in Section II. Our goal in this experiment is to investigate the effect of different filters on the input images for our model. To this end, we trained our model using the filtered images from filters that were described in II. Moreover, we also tested the effect of mix-up methods when they applied to the images along with the filters. Table IV illustrates our findings for a model trained with various filters and mix-ups with and without the DAM method. For the models with DAM, the training ϵ parameter is sampled from a normal distribution (μ = 0.35, σ = 0.08). In addition to this, we were eager to know how robust these models are. To measure the robustness of these models, we used input data with varying test ϵ between 0.1 and 0.5. Figure 8 illustrates our findings. The results indicates that the environmental filters acts better in smaller epsilon while the attention filter is not performing well which may be due to the omission of some part of the image. We will try to justify this behaviours later in the discussion section IV.

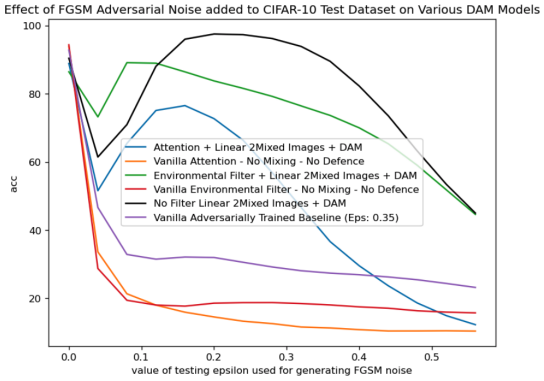

Fig. 8: Behaviour of different DAM models trained with different filter both in presence and absence of mixing functions on different extents of noise. In all DAM models the training ϵ has been sampled from Normal Distribution (μ = 0.35, σ = 0.08)

IV. Discussion

In this section we try to explain the behaviour of models observed in our experiment and also suggest further investigations for some of the results and future work. In our first experiment, the objective was to determine whether DAM is improving model performance. The results in table I indicate that mix-up improves the accuracy 0.6% and the robustness about 6.61%. However, DAM diminishes the accuracy for clean data by 6.36% but for the noisy data outperforms all the other approaches tremendously between 54.1%-68.63%. To the authors, it seems like a reasonable trade-off to be used in real-world applications which are most of the time noisy. To explain this performance on the noisy images, we have to point that the DAM method trains the model on noisy data so it makes the model more robust as is also depicted in Figure 5. In our second experiment, we see that training model with different ϵ causes a peak in the performance around the training ϵ. Our initial idea is that since the model observes more images with noises around those peaks, it learns how to classify the images better in that area. However, there are two

TABLE IV: Testing the effect of two proposed filters on the performance of a model in presence and absence mixup functions. DAM models is (Testing ϵ = 0.35) on CIFAR-10

	Functions		**clean input**	**Adversarial input (ϵ=0.25)**
Attention	*linear*	✓	88.86%	64.54%
Attention	*No Mixup*	✗	93.80%	13.13%
Environmental	*Linear*	✓	86.41%	**81.06%**
Environmental	*No Mixup*	✗	**94.32%**	18.80%

cases that this justification does not apply to. First, when the ϵ is small (in our case for ϵ=0.05) this peak does not happen. The second case is the applying of DAM without any mix-up, which does not see this peak in the performance. Our results indicate that the epsilon and mix-up together are causing this peak in performance but to extract the definite mechanism of this behaviour further investigation seems necessary. In the third experiment, we had two interesting observations: 1) The mixture of two images performs better in comparison to the mixing of three images, and 2) In different mix-up functions, horizontal mix-ups outperforms other functions. To explain the first observation, one reason could be the lack sufficient pixels from each image to learn the distinguishable features for each class. One experiment that could verify this hypothesis is to test this experiment on images with more pixels and higher qualities. For the former observation, the only justification we find is that in comparison with linear mix-up, horizontal mix-up contains more information since the blurriness of the image might deteriorate the feature extraction ability of our model. As for the last experiment, the results were not what we expected for the attention filter. Not only did the attention filter not help the model predict the classification, but diminished the performance of the model. One explanation for this decline in performance could be the fact that our model benefits from the surrounding objects when classifying the target object itself. For example, being in a metropolitan background makes the object more probable to be a car than a wild animal. This could also be the reason for the descending graph of accuracy in Figure IV. The figure also shows that that the environmental filter outperforms the attention filter, which confirms that the reason for under performing attention filter could be the omission of object surroundings.

V. Conclusion and Future Works

This project has tremendous potential in all of the modules addressed. In attention filters, only the core part of the target object is given higher importance; an approach to give equal priority to the peripherals of the object might improve results. Regarding the environmental filters, in this work a random normalized surrounding augmentation is used, but customizing the environment filter for the object could help.In this study, we have performed one linear mix-up and two non-linear mix-ups; multiple other mix-ups included in the work by Summers and Dinneen like F-mix, random square, VH-mix-up, random columns, random rows etc. can be explored for better results. Here, we have performed adversarial defense using TRADES algorithm, multiple other adversarial techniques can be leveraged to study the importance of the defensive approaches with respect to mixed-up clean and noisy data.

In this study, we aimed to enhance the robustness and improve the performance of the neural network by performing two variants of filters, three types of image mixings (between 2 different images as well as 3 different ones) and defensive adversarial training. We experimented using the different combinations of the above mentioned modules on clean data for two types of data: original CIFAR-10 and noisy CIFAR-10 (adversarial attack). When experiments were performed on the clean dataset, the accuracy was maintained in most of the mix-ups; but when performed on the noisy dataset, the accuracy significantly improved from 90.38% to 97.10% while using linear mix-ups in the DAM model. Although the filters didn't produce significant benefits, accuracy was still maintained and there is potential to perform efficient filtering techniques. Therefore, this project has provided an end to end enhancing approach for image classification tasks, and as proved to be efficient in multiple experimental scenarios.

References

[1] Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Nature, 542(7639), 115–118. https://doi.org/10.1038/nature21056

[2] Gulshan, V., Peng, L., Coram, M., Stumpe, M. C., Wu, D., Narayanaswamy, A., Venugopalan, S., Widner, K., Madams, T., Cuadros, J., Kim, R., Raman, R., Nelson, P. C., Mega, J. L., Webster, D. R. (2016). Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs. JAMA, 316(22), 2402–2410. https://doi.org/10.1001/jama.2016.17216

[3] Han, Z., & Xu, A. (2021). Ecological evolution path of smart education platform based on deep learning and image detection. Microprocessors and Microsystems, 80, 103343.

[4] Hosseini, M., Powell, M., Collins, J., Callahan-Flintoft, C., Jones, W., Bowman, H., & Wyble, B. (2020). I tried a bunch of things: The dangers of unexpected overfitting in classification of brain data. Neuroscience Biobehavioral Reviews.

[5] Dong, Y., Fu, Q. A., Yang, X., Pang, T., Su, H., Xiao, Z., & Zhu, J. (2020). Benchmarking adversarial robustness on image classification. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 321-331).

[6] Goodfellow, I. J., Shlens, J., Szegedy, C. (2014). Explaining and harnessing adversarial examples. arXiv preprint arXiv:1412.6572.

[7] C. Summers and M. J. Dinneen, "Improved Mixed-Example Data Augmentation," 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), Waikoloa, HI, USA, 2019, pp. 1262-1270, doi: 10.1109/WACV.2019.00139.

[8] Hawkins, D.M. The problem of overfitting. J. Chem. Inf. Comput. Sci. 2004, 44, 1–12. [CrossRef] [PubMed]

[9] Manu Goyal, Saeed Hassanpour , Moi Hoon Yap, Member, IEEE. Region of Interest Detection in Dermoscopic Images for Natural Data-augmentation. DOI: 10.13140/RG.2.2.16602.64966

[10] Alexander Buslaev 1 , Vladimir I. Iglovikov 2 , Eugene Khvedchenya 3 , Alex Parinov 4 , Mikhail Druzhinin 5 and Alexandr A. Kalinin. Albumentations: Fast and Flexible Image Augmentations. Information2020,11(2) , 125; doi: 10.3390/info11020125

[11] He, K., Zhang, X., Ren, S., Sun, J. (2016). Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 770-778).

[12] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A. (2016). Learning deep features for discriminative localization. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 2921-2929).

[13] Chorowski, J., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y. (2015). Attention-based models for speech recognition. arXiv preprint arXiv:1506.07503.

[14] Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., ... Bengio, Y. (2015, June). Show, attend and tell: Neural image caption generation with visual attention. In International conference on machine learning (pp. 2048-2057). PMLR.

[15] Yuan, Y., Sharoff, S. (2020). Sentence level human translation quality estimation with attention-based neural networks. arXiv preprint arXiv:2003.06381.

[16] Heo, J. (n.d.). Tootouch/whitebox-part1. Retrieved April 15, 2021, from https://github.com/TooTouch/WhiteBox-Part1

[17] Zhang, H., Cisse, M., Dauphin, Y. N., Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412.