Electronics and Computer Science
Faculty of Engineering and Physical Sciences
University of Southampton

Yiwei Liao
yl38u22@soton.ac.uk
April 2025

Enhancing Model Performance: The Impact of Data Augmentation Techniques in Machine Learning

Project supervisor: *Prof. Jonathan Hare*
Second examiner: *Prof. Age Chapman*

A final report submitted for the award of
Bachelor of Science

1 Abstract

Mixed Sample Data Augmentation (MSDA)is a class of techniques that combine two or more data samples to create a new synthetic sample[1]. Semantic segmentation (also known as pixel-level classification) is the computer vision task of assigning precise categorical labels, such as dog, cat, or goldfish, to every individual pixel in an image. The current MSDAs mostly focus on using a linear pixel mixing function (such as mixup) to linearly combine 2 images into a mixed image by following a mixing factor, or using a randomly generated area as the mixing mask when mixing (cutmix,fmix, etc.). However, all these methods are limited due to the uncertainty of each image, more specifically, the mask randomly generated might accidentally miss some important features of the original image, resulting in a hybrid synthetic image without any useful information from the original inputs.

Imagine, when performing MSDA, if we can choose to only use the area that has a valid semantic meaning, instead of mixing meaningless areas without any useful object information, this will force the model to simultaneously recognize areas with useful information about the main object of each input image used for generating hybrid image, resulting a more reliable and solid model when performing image classification tasks on different datasets.

This project aims to enhance the performance and robustness of image classification models by exploring new, advanced data augmentation techniques. Specifically, it focuses on developing a custom mixed sample data augmentation strategy based on a semantic segmentation model that performs object detection on the input image and then generates hybrid images that contain both useful object information from the original input image for different image datasets.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

ECS Statement of Originality Template, updated August 2018, Alex Weddell aiofficer@ecs.soton.ac.uk

2 Aknowledgements

I sincerely thank my supervisor, Prof. Jonathan Hare, for his invaluable guidance during our weekly meetings and insightful feedback on my progress report. His advice significantly improved my evaluation protocol, enabled me to streamline my report by eliminating unnecessary details, and fostered a rigorous yet accessible academic writing style that effectively balances conciseness, depth, and clarity. I would also like to thank my secondary examiner, Prof. Age Chapman. Her useful advice and guidance during our first meeting helped me form a well-structured final testing strategy and a clued evaluation process. I also want to express my profound gratitude to my parents for granting me the remarkable opportunity to pursue my studies and reside in the United Kingdom. Their unwavering financial and emotional support throughout my three years at the University of Southampton has provided me with an exceptional living and learning environment, enabling me to fully dedicate myself to the successful completion of my academic endeavors without distraction or concern. Last, I want to thank Hanqi Liao, a third-year student studying at Beijing University of Aeronautics and Astronautics, for sharing similar project experience, which helped me solve library conflicts during my training process.

Contents

3 Introduction

3.1 Background

Mixed Sample Data Augmentation (MSDA) has emerged as a pivotal technique for enhancing model generalization by synthesizing diverse training samples. Conventional methods such as *Mixup* [2], *CutMix* [3], and *FMix* [4] blend or replace image regions using linear interpolation, random patches, or Fourier-based masks. While these approaches improve robustness, their reliance on arbitrary spatial combinations often compromises semantic coherence. For instance, *Mixup* linearly interpolates entire images, indiscriminately mixing foreground and background [2], while *CutMix*'s rectangular masks risk splitting critical object features [3]. Recent analyses by Park et al. [1] emphasize that current MSDA methods prioritize mask diversity over semantic relevance, limiting their ability to preserve discriminative object regions.

3.2 Problem Statement

Despite their utility, conventional MSDA methods face a critical limitation:

- **Semantic Incoherence**: Randomly generated masks (e.g., *CutMix*'s squares or *FMix*'s Fourier patterns) sometimes bisect critical object regions, disrupt object integrity. And to the best of my knowledge, current MSDAs mostly focus on increasing mask diversity instead of paying attention to the type and content included in the generated mask: Does the mixing area contain useful information that can correctly represent the main object or label/class definition of that image? This results in the topic I'll mainly discuss in this project being undiscovered, and forms my motivation in doing this project.

3.3 Goals and Objectives

This project addresses these challenges by designing **Semantic Mix (Smix)**, a novel MSDA method that integrates semantic segmentation to guide hybrid image generation. The main objectives are:

- **Semantic-Aware Masking**: Replace random masks with segmentation-derived masks to preserve critical object regions 6.4.1.

By bridging semantic segmentation and data augmentation, this work advances context-aware augmentation strategies, as demonstrated by *Smix*'s state-of-the-art accuracy on the Tiny ImageNet dataset, shown in the later paragraph 6.7.

4 Report on Background Research and Literature Search

Mixed Sample Data Augmentation (MSDA) is a class of techniques that combine two or more data samples to create a new synthetic sample[1]. Unlike traditional augmentation methods that apply transformations to a single sample, MSDA methods leverage the inherent diversity of multiple data points, leading to more powerful augmentations. In recent years, MSDA has played a crucial role in enhancing the generalization ability of deep learning models, especially in computer vision tasks. Techniques, such as CutMix and Mix-up, have emerged as a powerful augmentation strategy. This literature review explores the development of different types of MSDA mask variants, lists and explains the theoretical basis, how they compare to other data augmentation methods, and how they will have a potential effect on the actual development of my own MSDA techniques.

4.1 Understanding Masks

If we want to know what exactly MSDA is, we need to understand the definition of the mask. In general, the design of the MSDA method can be referred to as designing a mask(the binary or probabilistic matrix used to control how and where the two input samples are combined). In MSDA, a mask determines which parts of the first input sample are preserved and which parts are replaced by corresponding regions from the second sample. Cut-out augmentation is a good example of the implementation of the binary mask, which has the value of 0 or 1, directly determining which sample contributes to each pixel. Different than the binary mask, a probabilistic or continuous mask may assign weights (e.g., values between 0 and 1) to softly combine the two samples. The combining operation can be defined as:

$$\tilde{x} = M \odot x_A + (1 - M) \odot x_B, \quad \tilde{y} = \lambda y_A + (1 - \lambda) y_B,$$

where $M \in \{0, 1\}^{W \times H}$ denotes a binary mask indicating where to drop out and fill in from two images[3]. Masks can also control the spatial layout of the augmentation. For example, A Gaussian mask in Gaussian Mixup gradually blends samples around a selected point using a Gaussian distribution. In Stochastic Mix-up, masks can introduce random patterns and increase the uncertainty of the blending samples. [5].

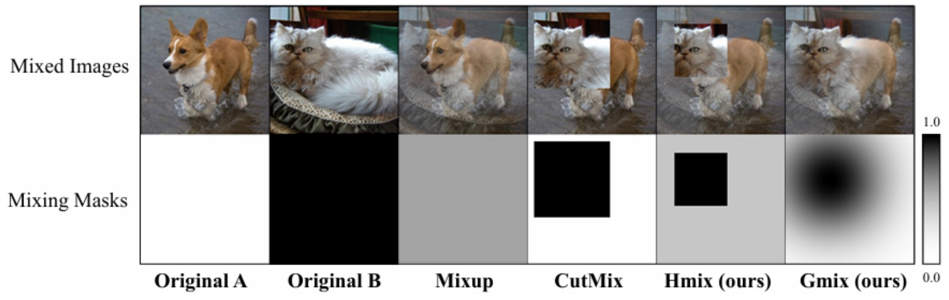

Figure 1: Examples generated by different MSDAs and masks[1]

4.1.1 Mixup

An important data augmentation method is the mixup training method, in which augmentation takes a linear combination of two different images[2]. So, for example, given an image of a cat and an image of a dog, it will combine the two images pixel-by-pixel through a weighted sum. The resulting image will contain, for example, the cat image at 70 percent pixel intensity combined with the dog image at 30 percent intensity. In this technique, the two input samples x_1 and x_2 along with their corresponding labels y_1 and y_2 are linearly interpolated to create a new synthetic sample x_{mixup}. Formally, this process can be represented as:

$$x_{\text{mixup}} = \lambda x_1 + (1-\lambda)x_2, \quad y_{\text{mixup}} = \lambda y_1 + (1-\lambda)y_2,$$

where $\lambda \in [0,1]$ is sampled from a Beta distribution, typically $\text{Beta}(\alpha, \alpha)$, and controls the interpolation ratio between the two samples[2]. The classification result is proportionally assigned based on this ratio. Mix-up data augmentation has been widely used in various fields since it can improve the model's robustness and generalization [1]. Introducing a mixture of samples prevents over-fitting and enhances the model's ability to generalize to unseen data. In 2021, the implementation of the mix-up data augmentation technique carried out by Özdemir and Sönmez improved its accuracy on the COVID-19 Dataset[6]. To visualize the mix-up process and for better understanding, here's an example of a mix-up operation of two COVID-19 Tomography images generated by Özdemir and Sönmez[6] using the COVID-19 Dataset provided by (Zhao et al., 2020[7]):

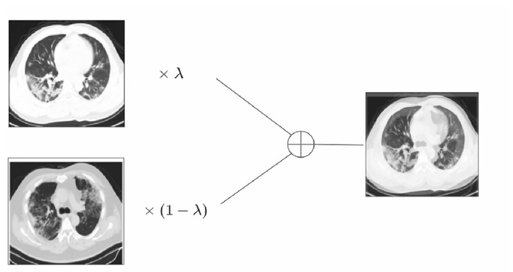

Figure 2: mixup operation [6]

4.1.2 Cutout

Cutout, proposed by DeVries and Taylor [8], is another data augmentation method designed to address the limitations of traditional augmentation. It works by randomly masking out square regions of an image during training, forcing the model to learn from partial views of objects [9]. This technique helps the model focus on the remaining visible portions, potentially learning more robust features. However, this method often leaves some parts of the image underutilized, which can be inefficient.

4.1.3 Cutmix

Cut-mix (Yun et al., 2019[3]) is another prominent mixed sample data augmentation (MSDA) technique that modifies the traditional image mixing approach by "cutting" and “pasting” patches from one image onto another. In this method, a rectangular patch is randomly selected from x_j, and it is then pasted onto x_i. The label for the new synthetic image is determined by the area of the patch from each image. Specifically, the new label is a weighted combination of the two labels, weighted by the relative area of the patch from each image. Like Cutmix, in CutOut, a random region of the sample is removed, but only filled with zero pixel values in the removed area. The classification result remains unchanged.

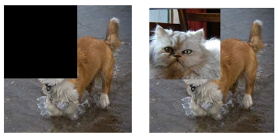

Figure 3: Cutout and Cutmix operation[3]

4.1.4 Hmix and Gmix

HMix is a hybrid approach that combines the methods of Mixup and CutMix, blending two samples using techniques from both.[1]It begins by cutting and pasting two samples in the style of CutMix, followed by linearly interpolating the values of the two samples as done in Mixup.[1]. Gmix stands for Gaussian Mixup, also introduced by (Park, Yun and Chun, 2022[1]) also integrates features of Mixup and CutMix. It selects a specific point and then gradually blends the two samples using a Gaussian function. The experiment carried out by (Park, Yun, and Chun, 2022) on CIFAR-100[10] and ImageNet[11] demonstrates that both HMix and GMix outperform state-of-the-art mixed sample data augmentation (MSDA) methods. It is not difficult to see that these MSDA techniques are based on the basic idea of mix-up and cut-out/cut-mix idea

4.1.5 Fmix

The success of mask-based MSDA methods such as CutMix stems from their ability to preserve the data distribution in a way that interpolation methods such as Mixup cannot. This is particularly beneficial in the perceptual space of CNNS, where convolutional neurons at specific spatial locations typically encode information from a single input at a time. This ”local consistency” ensures that spatially approximate elements within the mask come from the same data points.

Although effective, CutMix has limited mask diversity due to its reliance on square masks. FMix addresses this limitation by exploiting masks derived from low-frequency Fourier transforms, achieving a wider range of shapes and transitions while maintaining the desired local consistency. FMix is a variant of MixUp, CutMix, etc. It uses masks sampled from Fourier space to mix training examples[12]. it improves traditional mask methods by addressing their limitations and enhancing the effectiveness of data augmentation in neural network training. It maximizes the number of possible masks whilst preserving local consistency[4].

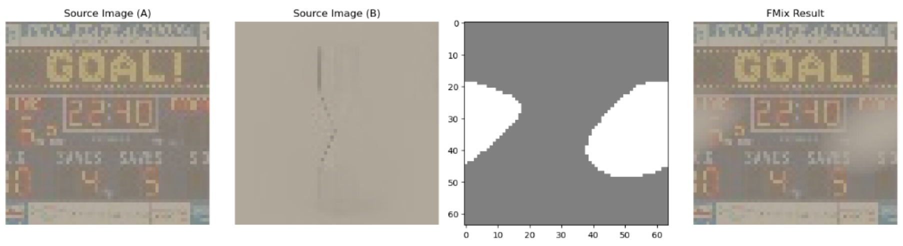

Figure 4: fmix example for TinyImageNet dataset

4.1.6 The limitation of the current method when generating masks

From the brief description for different masks above, it is not hard to see all of them either use random generated area as the mixing area or they chose to mix every pixels linearly, this can cause a problem, since the main object of the picture can be accidentally divided by the area chosen to be mixed, or the foreground and background might completely mixed and confused for the final output image. This will lead to a lack of enough information about the main object for the mixed image for final training.

For example, MixUp performs global blending on the entire image, resulting in mixed foreground and background, and the generated samples may be semantically incoherent (e.g., partial main object features are missing). The rectangular regions of CutMix are random and may cut into critical parts of the object (such as splitting the main object in two), resulting in semantic incompleteness. For fmix, the masks are based on the random generation of the Fourier transform and, while smooth, still random, and the mask quality is unstable: It sometimes failed generate the mask during my actual testing and training progress(the mask generated is empty for some reason and leads to to mixed image not having any kind of change at all) and the shape and coverage area randomness of the mask is large.

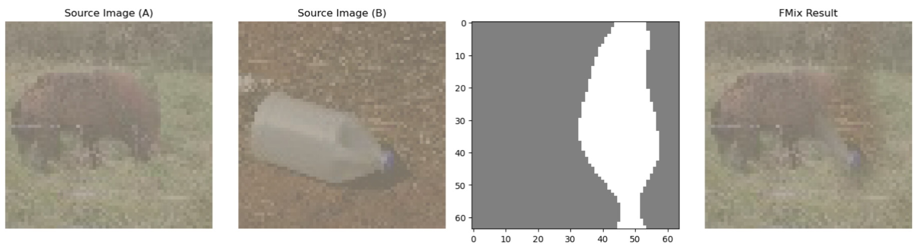

Figure 5: Where is the water bottle? Missing main object on the final mixed image for fmix

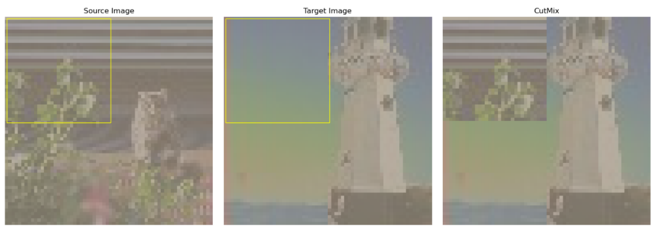

Figure 6: The missing cat. Missing main object on the final mixed image for cutmix

All these imperfections make me think of one question: Can we make the area chosen to be mixed meaningful?

4.2 Image Semantic Segmentation Method

The first idea that comes to my mind is image semantic segmentation, a method used to extract the main object of the image. The goal in many tasks is to extract regions that contain meaningful areas of the image, such as crops, urban areas, and forests of a satellite image[13]. It aims to assign a semantic class label (e.g., ”road” or ”pedestrian”) to each pixel in an image. In this field, multiple classical models have driven technological development through different innovations.

4.2.1 DeepLabv3

For example, DeepLabv3 (Chen et al., 2017)[14] proposed atrous convolution and atrous spatial Pyramid Pooling (ASPP) to capture multi-scale context information while maintaining feature resolution, often trained on PASCAL VOC [15], Cityscapes[16] and other datasets, has become the benchmark for universal scene segmentation. For the model being used later on my actual training (DeeplabV3 MobileNet-v3 segmentation model), it was pre-trained on a subset of the MS COCO [17] dataset using only the 20 categories that overlap with the PASCAL VOC 2012 dataset, plus a background class, resulting in a total of 21 classes (Detailed class categories is listed in appendix). Most importantly, the class categories are extremely similar to CIFAR-10[10] classes, making the DeepLabV3 segmentation model an ideal choice for semantically segmenting the CIFAR-10 input image data.

4.2.2 Mask R-CNN

Mask R-CNN (He et al., 2017) [18] focuses on instance segmentation, its improved RoIAlign technology and mask generation branch achieve high-precision object-level segmentation on MS COCO dataset.

4.2.3 FCN

FCN (Long et al., 2015)[19] achieved end-to-end pixel classification for the first time through full convolution and skip connection, laying the foundation for modern segmentation models.

4.2.4 U-Net

U-Net(Ronneberger et al., 2015) [20] for medical images performs well in small data tasks such as ISBI cell segmentation by virtue of a symmetric encoder-decoder structure and skip connection. Although achieving a good result on the cell dataset, but lacks the variety of classes of training images and similarity to my chosen training dataset.

4.2.5 PSPNet

PSPNet (Zhao et al., 2017) [21] aggregates the global context through the pyramid pooling module, which significantly improves the parsing ability of complex street view datasets such as Cityscapes.

4.2.6 Conclusion

These models push the boundaries of semantic segmentation techniques in different datasets (e.g., COCO for general scenes, Cityscapes for street View, BraTS[22] for medicine) through multi-scale feature fusion, context augmentation, or structural optimization, and provide core ideas for subsequent research.

4.3 The choice of the Image segmentation model

In this work, I'm using a pre-trained image segmentation model called **deeplabv3 mobilenet v3 large** with a MobileNetV3-Large backbone[23], and as I mentioned above, deepLabv3 with MobileNetV3 was pre-trained on a COCO dataset subset, covering the 20 PASCAL VOC 2012 categories plus a background class, totaling 21 classes. These categories closely align with CIFAR-10 classes, making DeepLabV3 a highly suitable segmentation model for processing CIFAR-10 image data for my initial experiment.

4.4 The choice of the training dataset

The choice of the training dataset for this project will be set to a relatively small-scale dataset mainly because of (1) I want to recreate the exact same accuracy analysis for my method with current existing methods by using the same datasets as they were used in most of the theses about MSDA. (2) Training on a large-scale image dataset requires a lot of GPU resources, and the time for training was hugely limited due to ECS GPU services were consistently occupied by some other students who were doing the same training just like me. These two facts led me to decide to choose 3 different datasets that satisfied the above condition during this experiment. The final training datasets were set to be CIFAR-10/100[24], Tiny-ImageNet[25], and probably more variants in the future.

CIFAR-10

Attribute	**Details**
Description	60,000 small images across 10 classes, 6,000 images per class[10].
Image Size	All images are 32 x 32 pixels with 3 color channels (RGB)[10].
Training set	50,000 images.
Test set	10,000 images.
Categories/Classes	Airplane, Automobile, Bird, Boat, Bottle, Bus, Car, Cat, Deer, Dog.

Table 1: Overview of the CIFAR-10 Dataset[10]

CIFAR-100

Attribute	**Details**
Description	100 classes with 600 images per class.
Image Size	32 x 32 pixels with 3 color channels (RGB)[10].
Training set	50,000 images.
Test set	10,000 images.
Categories/Classes	Divided into 20 superclasses, each containing 5 subcategories.
Example Superclasses	Vehicles: Bus, Car, Motorcycle, Bicycle.

Table 2: Overview of the CIFAR-100 Dataset[10]

Tiny-ImageNet

Attribute	**Details**
Description	subset of the ImageNet dataset,200 classes with 500 images per class.
Image Size	All images are 64 x 64 pixels with 3 color channels (RGB).
Training set	100,000 images.
Validation set	10,000 images.
Test set	10,000 images.
Categories/Classes	200 classes, each with 500 images distributed among various categories.

Table 3: Overview of the Tiny ImageNet Dataset[25]

5 Brief analysis and justification of the general solution to the problem

5.0.1 Overall design

In this project, my overall task is to develop a new MSDA method called Smix and verify the result by training ResNet-18 models on different datasets with the custom data augmentation technique. Smix uses semantic masks generated by a pre-trained segmentation model as the mixing area to mix images, aiming to enhance model performance by incorporating semantic information during training.

5.0.2 Why MSDA-Like object-based Augmentation?

The decision to develop a custom MSDA object-based augmentation technique stems from its potential to address the limitations of traditional augmentation strategies, such as Mix-up. Traditional approaches mainly rely on simple transformations that, while effective in improving generalization, may not fully exploit the potential diversity of augmented data. Custom MSDA-like augmentation algorithms introduce more advanced techniques, such as object layout, that can:

1. Increased data diversity: By incorporating objects extracted from external datasets or segmentation models, augmented samples can represent changes closer to the real world, enhancing the generalization ability of the model.
2. Improved robustness: Advanced object-based augmentation can expose models to more challenging and diverse data distributions, reducing sensitivity to specific patterns in the original dataset.
3. Outperforming traditional methods: Unlike linear interpolation, such as Mixup, object-based augmentation provides more fine-grained control over the placement and mixing of features, potentially resulting in richer and more informative samples.

5.0.3 Why use pre-trained segmentation models?

Ideally, if we can also train an image segmentation model for each dataset we use for doing the image classification task, this will fit each dataset the best. However, not like MS COCO[17], PSACAL VOC[15] and every other dataset used for training image segmentation model which has a folder contain information about the attributes of the main object, the dataset we chose (will explain in detail in the later paragraph) doesn't have pre-labeled area for training semantic segmentation models, which means if we want to train a unique model for each different dataset, we have to manually label the areas which contain the main object and this requires huge amount of workload (I tried to use a tool called labelme[26] to manually label the area for training image segmentation model on Tiny ImageNet. I use 5 images from each class as to form my training dataset, and I eventually gave up due to imperfection of the final model performance and the huge amount of workload.

5.0.4 Designed approach and risk analysis

Below is a high-level analysis and specification of the solution, structured as a design outline before implementation.

First, Since I need to compare the Smix's leaning outcome with different existing MSDAs, it is important to single control variables during training, It is necessary to make sure the dataset being used, the number of epochs for each training, the batch size and the SEED value for training to be explicitly the same; the only difference should be the way to implementing MSDA.

Secondly, in terms of the size of input of the pre-trained segmentation model requires, it might be different then the size of the dataset being used for training Image classification models. Necessary image pre-processing and Transformation might required to make the size of the input image fits the size of the input which semantic segmentation model requires.

Thirdly, the pre-trained segmentation may failed on recognize a certain class or certain type of image since the CIFAR-10/100, TinyImageNet dataset have many different classes then PASCAL VOC, the actual dataset which the segmentation model was trained on, what to do if the model failed to generate mask for mixing (Using combined MSDAs like using fmix when the model fails to generate masks) and how to prevent or reduce the possibility of such case happen also need to be consider in the final design case.

Last but not least, In terms of the metrics decided to use during training, except model evaluation performance metrics like Standard deviation, confusion matrix, and standard test accuracy comparison. It might also be worth taking the time required for training for each method into account since we don't want a method get a 2% improvement but require 300% more time for training.

6 Experiments and Detailed design

6.1 Advance notice

In this section, all listed designs as well as completed experiments are listed in chronological time order of completion (e.g., By the time I get the result after finishing experiment 1, I still haven't started designing any part of my actual smix). The results listed at the end of each experiment are the results that were available at the time of the completion of the experiment, so there are repetitions with the following chapter on the analysis of experimental results. However, I still list the results and summarize the reflection obtained from each experiment at the end of each experiment section, so as to let the reader better understand my thoughts by the time I complete each experiment.

6.2 Parameter used for all experiments

The parameters and the environment used for training are listed in detail in the appendix of this report.

6.3 Experiment 1: Initial image classification model training on CIFAR-10 Dataset

This experiment trains multiple image classification models using the RESNET-18 neural network on the CIFAR-10 data set. Existing methods such as mix-up,cut-mix, and f-mix are being used. The models were trained with different seeds and compared the average proportions of incorrect predictions with the model trained on the same dataset but without using any data augmentation. The model trained will be used for later comparison with the model trained by using s-mix.

Table 4: The image classification accuracy for mix-up, cut-mix, and f-mix, performed on the CIFAR-10 Dataset

Method	**Mean (%)**	**Std Dev (%)**	**Runs**	**Min (%)**	**Max (%)**
fmix	**94.83**	0.27	5	94.36	95.01
mixup	92.57	0.17	5	92.39	92.84
vanilla	89.96	0.27	5	89.69	90.38
cutmix	93.16	0.07	5	93.07	93.25

From this experiment, we can conclude that The model trained using MSDAs training makes predictions better than the model trained without MSDAs.

6.4 The Design of Smix

My proposed Semantic Mix (SMix) augmentation strategy integrates semantic segmentation with mixed-sample data augmentation to enhance model generalization. The design framework operates through four stages: (1) semantic mask generation, (2) image mixing, (3) Loss function design, and (4) Boundary case analysis, as detailed below.

6.4.1 Semantic Mask Generation

In order to detect the correct main object of the input image, a semantic mask needs to be generated, and ideally this needs to be done before the actual training loop and stored in a location so when it starts to do the actual training in the future, it will not generate the same masks again for every different seeds, which will slow down the training process due to doing unnecessary repetition works. In this case, for each training image $\mathbf{x}_i \in \mathbb{R}^{32\times32\times3}$ in the training dataset (CIFAR-10 in this example), a semantic mask $\mathbf{M}_i \in [0,1]^{32\times32}$ is precomputed using a DeepLabV3-MobileNetV3 (or any other model) segmentation model and stored in a location for future usage. The process involves the following steps:

1. Resize $\mathbf{x}_i$ to 64×64 or 32×32 or other sizes based on the training dataset and normalize using dataset statistics (For keeping the formula brief I'll use Tiny ImageNet as an example input dataset for the following explanation):

$$\tilde{\mathbf{x}}_i = \text{Resize}(\mathbf{x}_i), \quad \hat{\mathbf{x}}_i = \frac{\tilde{\mathbf{x}}_i - \boldsymbol{\mu}}{\boldsymbol{\sigma}},$$

where $\boldsymbol{\mu} = [0.485, 0.456, 0.406]$ and $\boldsymbol{\sigma} = [0.229, 0.224, 0.225]$.

2. The segmentation model f_{seg} outputs a probability map:

$$\mathbf{S}_i = f_{\text{seg}}(\hat{\mathbf{x}}_i) \in \mathbb{R}^{64\times 64\times 21},$$

   with raw class predictions $\mathbf{M}_{\text{raw}} = \arg\max(\mathbf{S}_i)$. In this case, a pretrained DeepLabV3 model predicts pixel-wise class labels (e.g., "cat", "dog", "sky"):

$$\mathbf{M}_{\text{raw}} = \arg\max(f_{\text{seg}}(\mathbf{x}_i))$$

   where f_{seg} denotes the segmentation model.

3. Mask Refinement: Raw segmentation outputs often contain jagged edges and holes. We apply four sequential operations:

   (a) *Gaussian Blurring*: Soften edges using a Gaussian kernel ($\sigma = 1$):

$$\mathbf{M}_{\text{blur}} = G_\sigma * \mathbf{M}_{\text{raw}}, \quad G_\sigma(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   (b) *Adaptive Thresholding*: Convert to binary mask using local intensity thresholds:

$$\mathbf{M}_{\text{thresh}}(x, y) = \begin{cases} 1 & \text{if } \mathbf{M}_{\text{blur}}(x, y) > \mu_{\text{local}}(x, y) + 2 \\ 0 & \text{otherwise} \end{cases}$$

   where μ_{local} is the mean intensity in a 5×5 neighborhood.

   (c) *Morphological Opening*: Remove small noise particles using:

$$\mathbf{M}_{\text{open}} = \text{Dilate}(\text{Erode}(\mathbf{M}_{\text{thresh}}))$$

   with a 3×3 elliptical structuring element.

   (d) *Contour Filling*: Ensure mask continuity by:

   - Identifying the largest connected component
   - Filling interior holes
   - Resizing to original dimensions:

$$\mathbf{M}_{\text{final}} = \text{Resize}(\text{Fill}(\text{LargestContour}(\mathbf{M}_{\text{open}})), (32, 32))$$

The refined mask $\mathbf{M} \in \{0, 1\}^{32\times 32}$ satisfies:

- $\mathbf{M}(x, y) = 1$ for object regions
- $\mathbf{M}(x, y) = 0$ for background

This continuous binary mask enables precise object-aware mixing in subsequent processing stages.

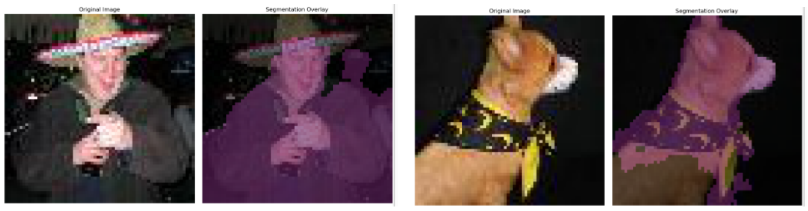

Figure 7: The semantic mask(The pink region) generated for TinyImageNet Dataset using DeepLabV3-MobileNetV3

6.4.2 Semantic-Guided Image Mixing

The mixing process aims to combine pairs of images using semantic masks as mixing guides. This can be described as the following steps:

1. Random Pairing: For a batch of images $\{\mathbf{X}_i\}_{i=1}^{B}$, generate shuffled indices via permutation:
$$\text{Indices} = \text{Permute}([1, 2, ..., B]) \tag{1}$$
Each image $\mathbf{X}_A$ pairs with $\mathbf{X}_C = \mathbf{X}_{\text{Indices}[A]}$.

2. Mask-Guided Blending: Combine images using element-wise convex combination:
$$\mathbf{X}_{\text{mix}} = \alpha \mathbf{M}_A \odot \mathbf{X}_A + (1 - \alpha \mathbf{M}_A) \odot \mathbf{X}_C \tag{2}$$
where:
   - $\mathbf{M}_A \in [0, 1]^{H \times W}$: Semantic mask for image A
   - $\odot$: Hadamard (element-wise) product
   - $\alpha \in [0, 1]$: Blending strength (empirically set to 0.4)

   The mixing process creates hybrid images where:
   - Object regions ($\mathbf{M}_A = 1$) preserve α-weighted features from $\mathbf{X}_A$
   - Background regions ($\mathbf{M}_A = 0$) fully adopt features from $\mathbf{X}_C$

   Here is an example, consider $\mathbf{X}_A$ containing a human and $\mathbf{X}_C$ containing a goldfish:
$$\mathbf{X}_{\text{mix}}(x, y) = \begin{cases} 0.4\mathbf{X}_A(x, y) + 0.6\mathbf{X}_C(x, y) & \text{if } (x, y) \in \text{human body} \\ \mathbf{X}_C(x, y) & \text{otherwise} \end{cases}$$

   My theoretical foundation of semantic-guided Image mixing rests on two critical mechanisms: (1) mitigating model over-reliance on global image statistics through region-specific feature disentanglement, and (2) establishing physiologically plausible transitions between hybrid regions via edge-aware blending with Gaussian-smoothed masks.

In short terms(take the example above), implementing smix will force the model to simultaneously recognize human body features in masked regions and goldfish body features elsewhere.

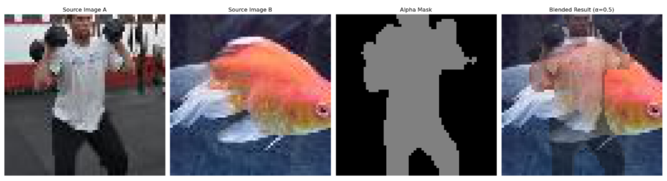

Figure 8: Visualization of the semantic mixing process of a human body to a goldfish, with $\alpha = 0.5$

6.4.3 Loss function

The loss function combines cross-entropy terms for both original labels. In smix, the loss function is defined as follows:

$$\mathcal{L} = \frac{1}{2N} \sum_{i=1}^{N} \left[\mathcal{L}_{\text{CE}} \left(f_{\text{model}}(\mathbf{x}_{\text{mix}}^{(i)}), y_A^{(i)} \right) + \mathcal{L}_{\text{CE}} \left(f_{\text{model}}(\mathbf{x}_{\text{mix}}^{(i)}), y_C^{(i)} \right) \right] \tag{3}$$

where:

- $y_A^{(i)}$: Label of primary image $\mathbf{x}_A^{(i)}$
- $y_C^{(i)}$: Label of shuffled image $\mathbf{x}_C^{(i)}$
- $\mathcal{L}_{\text{CE}}$: Cross-entropy loss $-\sum y \log p(y|\mathbf{x})$

6.4.4 Boundary Case Analysis

When $\alpha = 1$, the mixing reduces to:

$$\mathbf{X}_{\text{mix}} = \mathbf{M}_A \odot \mathbf{X}_A + (1 - \mathbf{M}_A) \odot \mathbf{X}_C \tag{4}$$

The loss enforces:

- **Foreground**: 100% supervision for class y_A in masked regions
- **Background**: 100% supervision for class y_C elsewhere

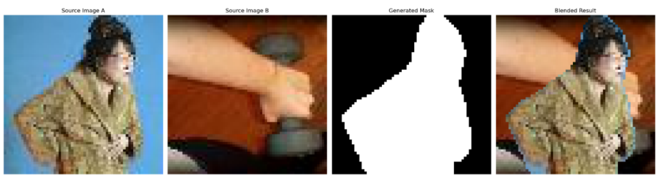

Figure 9: Visualization of semantic mixing process when $\alpha = 1$. (a) Original human image, (b) background image, (c) Semantic mask generated, (d) Generated hybrid image with human body preserved.

6.4.5 Theoretical Motivation

The smix design introduces three fundamental constraints:

1. **Region-Specific Supervision**:

$$f_{\text{model}}(\mathbf{x}_{\text{mix}})(x,y) \approx \begin{cases} f_{\text{model}}(\mathbf{x}_A)(x,y) & \text{if } \mathbf{M}_A(x,y)=1 \\ f_{\text{model}}(\mathbf{x}_C)(x,y) & \text{otherwise} \end{cases} \quad (5)$$

2. **Contextual Disambiguation**: Forces differentiation between local features (e.g., cat ears vs dog fur) rather than global context
3. **Gradient Balancing**: Equal weighting prevents dominance by either label during backpropagation

6.4.6 Potential Benefits for Smix

- Pre-trained models (e.g. DeepLabV3) that have learned general-object segmentation capabilities on large datasets (COCO, etc.) can be directly transferred to TinyImageNet (or any other dataset) without manually labeling masks.
- Generating masks for all training images in advance can avoid real-time calculation during training (otherwise, the segmentation model needs to be called for each forward propagation, which greatly increases the computational cost).
- Prevents overfitting to single-image statistics
- Encourages local feature learning
- Maintains stable training through symmetric loss design

6.4.7 Potential risks and imperfections for Smix

- The performance of smix hugely relies on pre-trained models, the pretrained segmentation model might not be able to detect the main object correctly for a small-sized image dataset(since it was trained on a dataset with much bigger image size and different classes). In some cases, it gets confused about foreground and background, and accidentally sets the main object as background and uses the actual background as a semantic mask for mixing.
- Smix requires mask precomputation and takes time to generate the mask, resulting in a slower training speed.

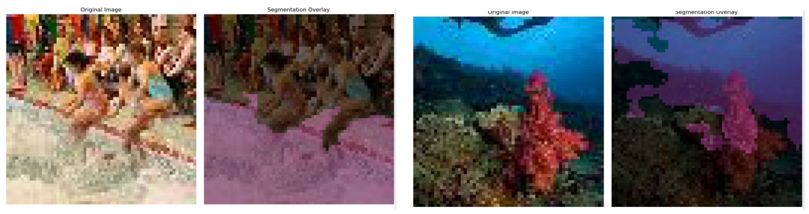

Figure 10: Failed cases when the semantic segmentation model failed to detect the main object (pink region is the mask generated)

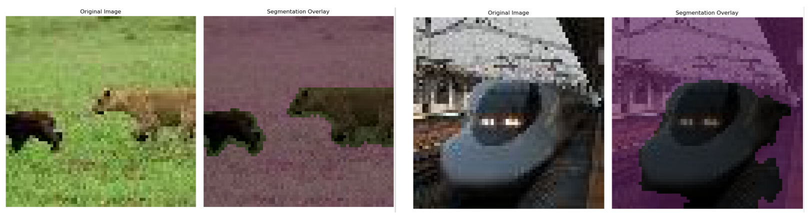

Figure 11: Failed cases when the semantic segmentation model takes the main object as the background (pink region is the mask generated)

6.5 Experiment 2: Training model with smix on CIFAR-10 and model accuracy analysis

In this experiment, I trained an image classification model using ResNet-18 on the CIFAR-10 dataset. The model was trained with the same parameters as it was used in experiment 1, but using smix instead. The result is shown as follows:

Table 5: The image classification accuracy for mix-up, cut-mix and f-mix, with Batch Size(128), 150 Epochs, learning rate=0.1 and mixup alpha value =0.2, training was performed on 5 different random seeds on the CIFAR-10 Dataset

Method	Mean (%)	Std Dev (%)	Runs	Min (%)	Max (%)
fmix	**94.83**	0.27	5	94.36	95.01
mixup	92.57	0.17	5	92.39	92.84
smix	93.66	0.29	5	93.40	94.03
cutmix	93.16	0.07	5	93.07	93.25
vanilla	89.96	0.27	5	89.69	90.38

Based on the result, for the image classification task on the CIFAR-10 dataset, fmix performs the best among all 4 data augmentation techniques, while smix performs second best, achieving a 93.66 percent mean accuracy in 5 runs. A more detailed Accuracy distribution map and Accuracy Trend distribution map are listed below:

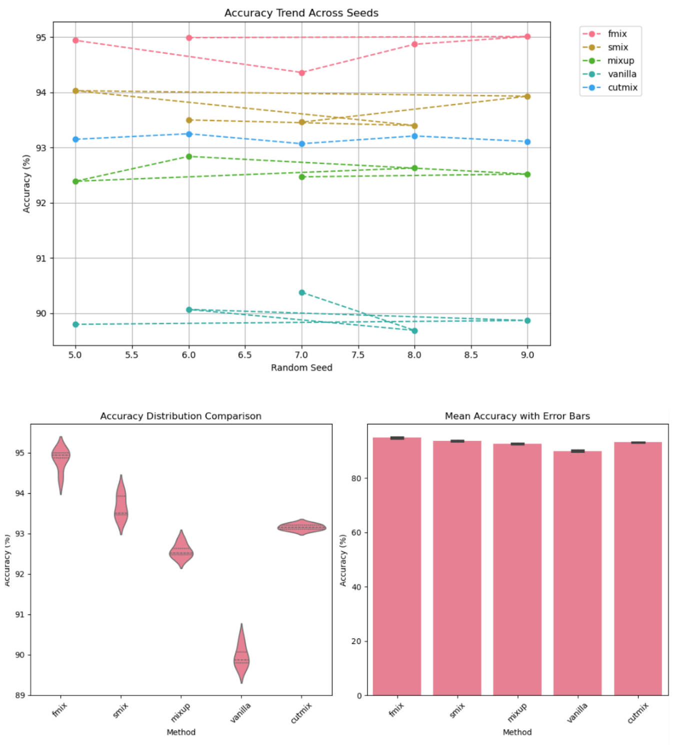

Figure 12: Accuracy distribution map and Accuracy Trend distribution map

From this experiment, fmix achieves the best result among all MSDA methods. I then discovered that smix mostly failed at predicting cat images and dog images by analyzing its confusion matrix.

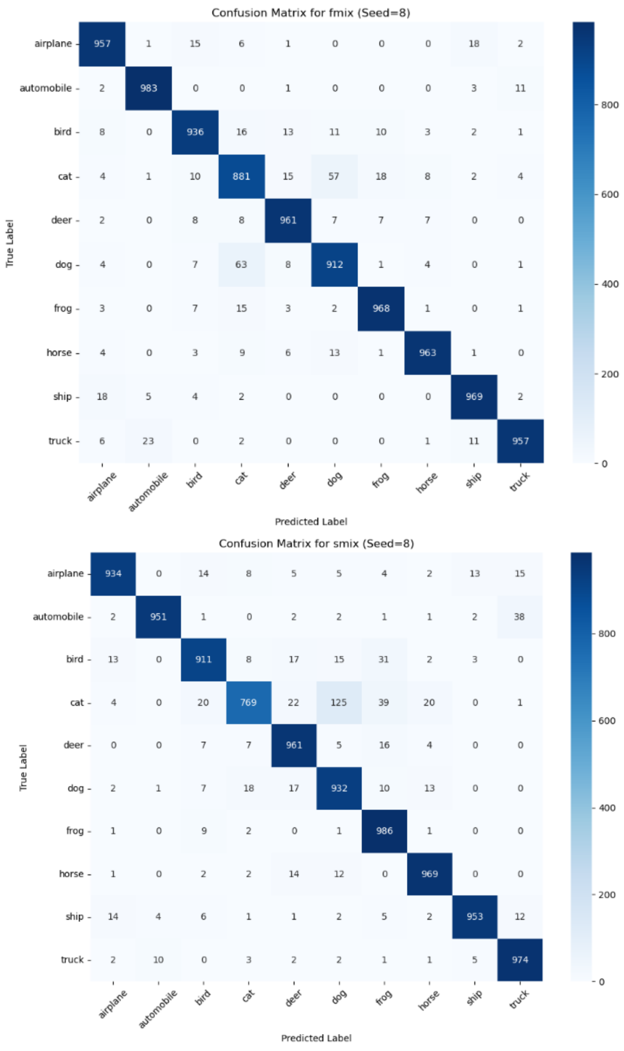

Figure 13: confusion matrix for smix and fmix on CIFAR-10 dataset

6.6 Experiment 3: Training models on CIFAR-100

In this experiment, I trained an image classification model using ResNet-18 on the CIFAR-100 dataset. The model was trained with the same parameters as it was used in experiment 1, but using smix instead. The result is shown as follows: From this experiment,

Table 6: The image classification accuracy on CIFAR-100 Dataset

Method	**Mean (%)**	**Std Dev (%)**	**Runs**
fmix	**75.26**	0.21	5
mixup	74.65	0.46	5
smix	71.60	0.29	5
cutmix	72.95	0.31	5
vanilla	70.28	0.22	5

fmix, again, showing dominance in small-scale input image

6.7 Experiment 4: Training models for TinyImageNet

In this experiment, I trained an image classification model using ResNet-18 on the TinyImageNet dataset. The parameter used and the training results are listed as follows:

Table 7: The image classification accuracy for TinyImageNet Dataset

Method	**Mean (%)**	**Std Dev (%)**	**Runs**
fmix	61.09	0.37	5
mixup	56.68	0.12	5
smix	**64.93**	0.05	5
cutmix	62.57	0.29	5
vanilla	55.28	0.44	5

6.8 Model Evaluation and Comparison

In this section, I'll mainly discuss and analyze the results from the above experiments. I'll briefly state the experiment results and then analyze the cause for the result and give possible improvements and optimization based on the fact I discovered from the result and analysis.

6.8.1 Image classification accuracy

Table 8: Image Classification Accuracy (%) Across Datasets

Method	**CIFAR-10**		**CIFAR-100**		**TinyImageNet**	
	Mean %	Std	Mean %	Std	Mean %	Std
fmix	**94.83** %	0.27	**75.26** %	0.21	61.09 %	0.37
mixup	92.57 %	0.17	74.65 %	0.46	56.68 %	0.12
smix	93.66 %	0.29	71.60 %	0.29	**64.93** %	0.05
cutmix	93.16 %	0.07	72.95 %	0.31	62.57 %	0.29
vanilla	89.96 %	0.27	70.28 %	0.22	55.28 %	0.44

Training details:

- CIFAR-10/100: 150 epochs, lr milestones at 50/100
- TinyImageNet: 200 epochs, lr milestones at 100/150
- Bold = best performance per dataset

6.8.2 Difference of time required for training model with different MSDAs

Since I managed to optimize my smix code by generating and storing semantic masks before the actual training starts, there was only a tiny increase in the time (a 30-minute training time increase on CIFAR-10/100 and a 2-3 hours training time increase on Tiny-Imagenet) required when training the model using smix. Under such conditions, it is not worth taking the time efficiency for training each model into account.

6.8.3 Model performance conclusion

From the table above, it is not hard to see that Fmix achieved the highest mean accuracy among CIFAR-10 (94.83 %) and CIFAR-100(75.26%). Smix performs the second best on CIFAR-10 (93.66%); however, it has the second worst performance on CIFAR-100 (71.60%), with only a 1.32 percent performance increase over vanilla (70.28%). For TinyImageNet, Smix achieves the best result of the MSDAs being tested so far, resulting in a 64.93% classification accuracy.

6.8.4 Potential causes for the underperformance of smix

The potential causes for the underperformance of SMix on Small-Scale Datasets are listed as follows:

- The dataset used to train the semantic segmentation model contains classes that do not match the classes of the training data. The dataset used to train Deeplabv3 (PASCAL VOC) only has 20 different classes, but CIFAR-100 has up to 100 classes. As a result, some image in CIFAR-100 cannot be well recognized by the semantic segmentation model.
- Smix's mask smoothing method might oversmooth discriminative features in small images, reducing the network's ability to learn sharp decision boundaries.
- SMix's performance may depend heavily on parameters like blending ratio thresholds or region granularity, which were not tuned here. Suboptimal defaults could disadvantage it on smaller datasets.
- The performance of smix hugely relies on pre-trained models, it could fail to detect the main object correctly for a small-sized image dataset since the segmentation model was trained on a dataset with a much larger image resolution. This will result in the semantic mask either being empty or with random noise, a detailed proof is listed below:

When semantic masks fail to be properly generated (i.e., all-zero masks or random noise masks), the mixing operation degenerates to:

$$\mathbf{X}_{\text{mix}} = \begin{cases} \alpha\mathbf{M}_A \odot \mathbf{X}_A + (1-\alpha\mathbf{M}_A) \odot \mathbf{X}_C & \text{Valid mask} \\ \mathbf{X}_C & \text{All-zero mask} \\ \text{Random pattern} & \text{Noisy mask} \end{cases}$$

This means the model receives completely shuffled images from batch pairs and loses all semantic guidance from the original images. The loss function then changed to the following:

$$\mathcal{L} = \frac{1}{2}\underbrace{\mathcal{L}_{\text{CE}}(\mathbf{X}_C, y_A)}_{\text{Mismatched term}} + \frac{1}{2}\mathcal{L}_{\text{CE}}(\mathbf{X}_C, y_C) \tag{6}$$

For Complete Mask Failure (All-Zero Masks) The mixing operation degenerates to:

$$\mathbf{X}_{\text{mix}} = \mathbf{X}_C \tag{7}$$

This creates a critical *label-feature mismatch* with three fundamental consequences:

- Input contains 100% features from image C:
- Loss enforces 50% supervision for label A:
- Effect: Induces contradictory learning signals comparable to 50% label noise:

$$\tilde{y} = \begin{cases} y_A & \text{with probability } 0.5 \\ y_C & \text{with probability } 0.5 \end{cases}$$

For Partial Mask Failure (Noisy Masks) This generates hybrid images with random blending patterns:

$$\mathbf{M}_A \sim \mathcal{U}(0,1)^{H\times W} \tag{8}$$

Resulting in:

- Non-semantic blending patterns:
- Equivalent to uncontrolled MixUp regularization:

$$\lambda \sim \mathcal{U}(0,1), \quad \mathbf{X}_{\text{mix}} = \lambda \mathbf{X}_A + (1-\lambda)\mathbf{X}_C$$

From the mathematical reasoning above, we can easily find that if the image segmentation model does not generate the correct mask, the blended image is really just another random image, but the loss function still requires the model to predict both labels simultaneously. This will create a label-feature mismatch where input contains 100 percent features from image C, but loss enforces 50 percent supervision for label A (for all zero masks). This can cause the model to learn the wrong features because the input and label do not match. The loss function is the average of the cross-entropy of the original and shuffled labels. If the mask is invalid and the blended image is independent of the original label, the first cross-entropy term (corresponding to the original label) will become meaningless because the input image has been replaced with another image, but the label is still the original. This will introduce noise and interfere with the learning of the model. Some potential improvements can be made, I'll mainly discuss that in a later section.

6.8.5 Verify the possible cause for model underperformance

To verify whether the performance of smix on small-scale image datasets is caused by the error when generating masks, I used TensorBoard to get a real-time visualization to monitor the masks generated for training. I also added a mask validation block by checking:

```
if torch.mean(mask) < 0.01:
```

The analysis revealed that approximately 50-60% of images in CIFAR-10/100 had zero identified main objects in their generated masks. Since there isn't any good way to tell whether a mask is a correct one or a noisy one without manually checking the mask shape, in this case, I'll just consider correct and noisy masks both as valid masks. Figure 14 shows the detailed comparison for all masks generated across different datasets:

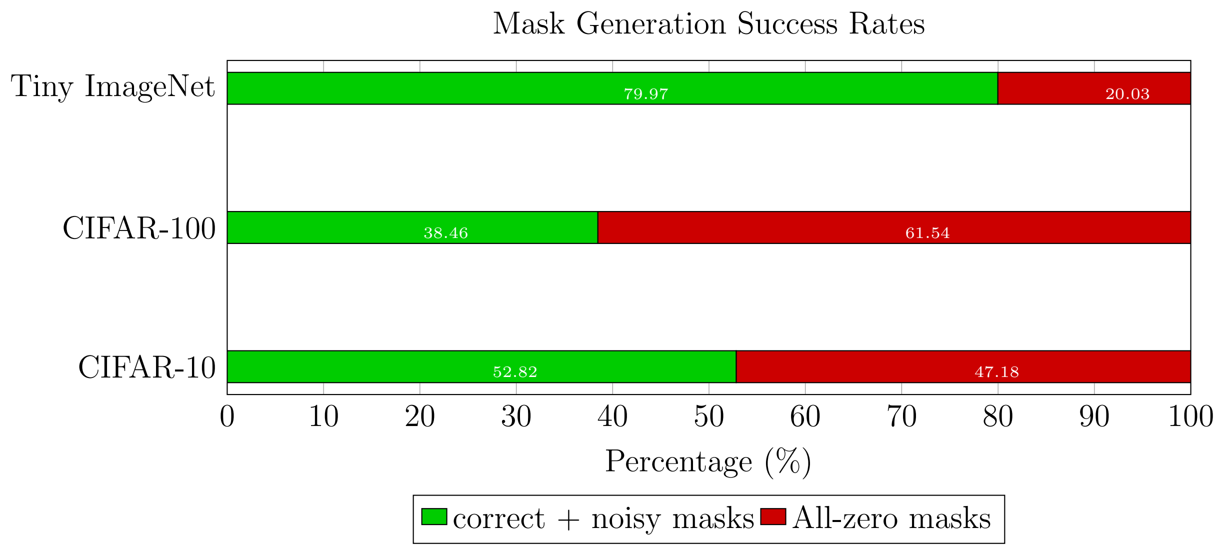

Figure 14: Distribution of mask generation results showing valid(not correct) versus invalid masks across datasets. The significant rate of invalid masks in CIFAR datasets explains SMix's bad performance on small-resolution image datasets.

This analysis demonstrates why SMix performs better on larger datasets like TinyImageNet (with 79.97% valid masks) compared to smaller datasets like CIFAR-10 (52.82% valid) and CIFAR-100 (38.46% valid). The high proportion of invalid masks in smaller datasets directly impacts the mixing effectiveness during training.

6.9 Possible Optimizations for Smix

Based on the experimental results and failure analysis, I propose the following optimization strategies to enhance Smix's performance across different datasets. These solutions address both technical limitations identified in mask generation and broader challenges in semantic-guided augmentation.

6.9.1 Segmentation Model Adaptation

Issue: The pre-trained DeepLabV3 model exhibits poor performance on small-scale datasets like CIFAR-10/100 due to resolution mismatches (32×32 vs. 513×513 input size) and class misalignment (PASCAL VOC vs. CIFAR classes)

Optimization Strategies:

- *Fine-tuning with Target Data:*
  - Employ lightweight architectures (e.g., MobileNetV3-Small) pre-trained on COCO
  - Fine-tune using a manually labeled subset of CIFAR-10/100 (100 images per class)
- *Resolution Adaptation:*
  - Upscale low-resolution images using ESRGAN [27]:

$$\mathbf{x}_{\text{HR}} = G_{\text{ESRGAN}}(\mathbf{x}_{\text{LR}}) \tag{9}$$

6.9.2 Hybrid Augmentation Strategy

Issue: Smix underperformed FMix by 3.66% on CIFAR-100 due to 61.54% mask failure rate (Section 6.8.5).

Optimization: Adaptive blending logic, using fmix instead when failed to generate semantic mask:

$$\mathbf{x}_{\text{mix}} = \begin{cases} \alpha\mathbf{M} \odot \mathbf{x}_A + (1-\alpha\mathbf{M}) \odot \mathbf{x}_B & \text{if valid}(\mathbf{M}) \\ \beta\mathbf{M}_{\text{FMix}} \odot \mathbf{x}_A + (1-\beta\mathbf{M}_{\text{FMix}}) \odot \mathbf{x}_B & \text{otherwise} \end{cases} \tag{10}$$

6.9.3 Dynamic Mask Validation combined with hybrid augmentation

Issue: Invalid masks (all-zero/noisy) accounted for 47.18% of CIFAR-10 cases , creating label-feature mismatches as analyzed in Equation 6.

Optimization Implementation:

Listing 1: Real-time Mask Validation

```
def validate_mask(mask, threshold=0.01):
    if mask.mean() < threshold or entropy(mask) < 0.5:
        return fmix(x_A, x_B)  # Fallback to FMix
    else:
        return semantic_mix(x_A, x_B, mask)
```

Metrics:

- *Entropy-based Filtering:* $H(\mathbf{M}) = -\sum p(m)\log p(m)$, threshold=0.5
- *Edge Density:* $\frac{\|\nabla\mathbf{M}\|_1}{HW} > 0.05$

6.9.4 Loss Function Refinement

Issue: The symmetric loss formulation in Equation 3 enforced equal weighting between labels y_A and y_C, even when semantic masks were invalid (e.g., $\mathbf{M}$ being all-zero or noisy).

In this case, I suggest an optimized loss function that can be dynamically adjusted based on mask validity. My rough ideas and thoughts are listed below.

My new loss function:

$$\mathcal{L} = w_A\mathcal{L}_{\text{CE}}(y_A) + w_C\mathcal{L}_{\text{CE}}(y_C) \tag{11}$$

where the weights w_A and w_C are dynamically adjusted based on mask validity:

$$w_A = \frac{1}{HW}\sum_{i,j}\mathbf{M}(i,j) \quad \text{(Normalized mask coverage)}$$

$$w_C = 1 - w_A \quad \text{(Complementary background weight)}$$

My new loss function design rationale

- *Mask Coverage as Confidence*: w_A quantifies the proportion of valid semantic regions in the mask $\mathbf{M}$. Higher w_A indicates stronger alignment between $\mathbf{x}_{\text{mix}}$ and y_A.

- *Adaptive Label Weighting*: When $w_A \to 0$ (invalid masks), the loss naturally transitions to $\mathcal{L} \approx \mathcal{L}_{\text{CE}}(y_C)$, suppressing contradictory supervision for y_A.

Implementation Details:

- *Normalization*: The $\frac{1}{HW}$ term ensures $w_A \in [0, 1]$, regardless of input resolution.
- *Edge Cases*:
  - For $\mathbf{M} = \mathbf{0}$, $w_A = 0 \Rightarrow \mathcal{L} = \mathcal{L}_{\text{CE}}(y_C)$
  - For $\mathbf{M} = \mathbf{1}$, $w_A = 1 \Rightarrow \mathcal{L} = \mathcal{L}_{\text{CE}}(y_A)$

7 Reflection

This project has provided valuable insights into the intersection of semantic segmentation and data augmentation, revealing both the potential and challenges of semantic-guided MSDA techniques. During my 6-month development cycle, I had a thorough experience on training models under the PyTorch framework, starting from pre-processing the dataset, designing my own augmentation method, to performing the actual training, then analyzing the model from different perspectives. This project significantly improved my ability in critical thinking, and some personal soft/hard skills were also developed and formed during this 6-month development cycle.

7.1 Technical Achievements

- **Semantic-Aware Augmentation Validity:** My hypothesis—that preserving object integrity improves augmentation quality—was validated on TinyImageNet, where Smix achieved state-of-the-art 64.93% accuracy, outperforming FMix by 3.84%. This confirms that context-aware mixing can enhance model generalization when segmentation reliability is ensured.
- **Failure Mode Quantification:** Through systematic mask analysis (Section 6.9.3), I established a direct correlation between mask validity and performance: I found that an increase in valid masks improved final testing accuracy.

7.2 Methodological Lessons

- **Pre-trained Model Limitations:** The 61.54% mask failure rate on CIFAR-100 exposed critical gaps in transfer learning for low-resolution domains, necessitating future work on resolution-robust architectures.
- **Smix imperfectness** The imperfections of Smix teach me an important lesson: always think as much as you can before designing the way to achieve your goal. In the design and planning stage, I should've considered the possible problems that might exist as much as possible. I did not consider the use of a combination of different MSDAs when one of them failed, which led to the shortcomings of existing methods.

7.3 Personal Development

- **Engineering Skills:** The project demanded proficiency in multi-stage training pipelines, including:
  - Training models on large GPU clusters set under a Linux environment.
  - Distributed training optimization on GPU clusters
  - Data pre-processing with OpenCV
  - Custom DataLoader integration in PyTorch

7.4 Collaboration Insights

- **Open Source Leverage:** Adapting the DeepLabV3 implementation from [14] saved around 40 development hours, highlighting the value of community-driven tooling. However, debugging compatibility issues between TorchVision 0.12 and CUDA 11.4 and my pending model training request on the ECS Iridis5 GPU cluster consumed unexpected time (15% of the project schedule).
- **Advisory Impact:** Weekly meetings with Prof. Hare and useful advise given from the feedback of my progress report not only helped refine my evaluation protocol, but also helped me to filter out unimportant parts of my report and helped me cultivate a rigorous academic writing style that balances conciseness with depth, ensuring precision while remaining accessible to both specialized and general audiences.

8 Project Managment

To ensure the successful execution and timely completion of this project, I implemented a variety of strategies tailored to planning, organization, and communication needs.

8.1 Project planning and Time Management

For scheduling and tracking progress, I divided the project into distinct phases: background research, algorithm development, implementation, experimentation, and report writing. Each phase was assigned approximate deadlines, which I monitored using the Gantt chart feature on Lucidchart. This allowed me to keep track of my to-do lists, ongoing tasks, and completed milestones. I also updated my actual planning based on the changing situation (e.g., supervisor advice and feedback, challenges and difficulties faced during my actual code implementation). More specifically, I encountered difficulties in handling conflicts between different libraries and multiple terminations of my training process and I also performed a more thorough overview of the project by doing more literature review, which led to new tasks added to my to-do list during my project design cycle. This results in a slightly different time-planning of my project compared with my original plan, as shown in the Gantt chart below. In general, using a Gantt chart helps me stay organized and motivated, ensuring that key objectives are met on schedule.

8.2 Documentation and Organization

To manage my project materials efficiently, I utilized Git for version control of my code, enabling me to track changes, revert to previous versions when necessary, and maintain a clean and reliable codebase. For documentation, I relied on Google Docs and Jupyter Notebook to store research notes, experiment logs, and perform result analysis on the trained model. Especially Jupyter Notebook. This tool provides me with a powerful analysis environment and instant visualization of the hybrid images and masks generated. Most of the demonstration images for different cases were generated by using Jupyter Notebook.

8.3 Communication and Feedback

Although my project was largely independent, I maintained regular communication with my supervisor, Prof. Jonathan Hare, through Microsoft Teams updates and weekly briefing meetings. For each meeting, I prepared concise summaries of my progress, challenges encountered, and next steps. These updates fostered productive discussions, allowing me to receive timely feedback and guidance that kept the project aligned with its goals. This approach also helped refine my evaluation protocols and writing style, as highlighted in 7.4

8.4 Additional Monitoring

To further enhance productivity, I maintained a daily log of activities and time allocation. This practice enabled me to reflect on my workflow, identify inefficiencies, and adjust my schedule as needed to stay on track.

Overall, these project management techniques provided a structured framework that supported the efficient development of the Semantic Mix (Smix) augmentation strategy, ensuring that deadlines were met and the project objectives were successfully achieved.

9 Conclusions and Future Work

9.1 Limitations and Current Findings

Due to time constraints, my evaluation of Smix was limited to a subset of datasets and model architectures. My experimental results reveal several key insights:

- **Dataset-Specific Performance Variance:** While Smix achieved competitive results on Tiny-ImageNet, it underperformed FMix on CIFAR-100. This suggests Smix's effectiveness depends heavily on dataset characteristics and model capacity.
- **Untested Combinations:** Different model architectures like Pyramid-Net[21] and WRN(Wide Residual Networks)[28] remain invalidated, along with datasets such as Fashion MNIST[29], Google commands[30](As these dataset already being tested by Harris et.al [4] which makes future training results can be easily checked and compared) caltech-101[31] is also an ideal dataset for further experiments.

9.2 Future Research Directions

To address these limitations and enhance Smix's generalizability, I propose the following research agenda:

9.2.1 Architectural and Dataset Expansion

- **Model Diversity:** Validate Smix on:
  - Lightweight architectures (ResNet-10, MobileNetV3)
  - High-capacity models (VGG-16, PyramidNet)
  - Non-CNN architectures (Vision Transformers)
- **Dataset Coverage:** Extend evaluation to:
  - Speech datasets (Google Commands V2[30])
  - Fashion MINIST[29]
  - Medical imaging (BraTS[22])

9.2.2 Using self-trained segmentation model

Using a self-trained segmentation model on a manually labeled subset of the input training dataset used to train the image classification model (ideally 50-100 images per class for CIFAR-10).

9.2.3 Implement Hybrid MSDA Strategies

Develop adaptive blending mechanisms combining Smix's semantic awareness with existing MSDA strengths, as described in6.9

The Gantt chart shows the schedule of both planned and actual timelines for completing this project during the whole project working period(Oct 14, 2024, to May 2025).

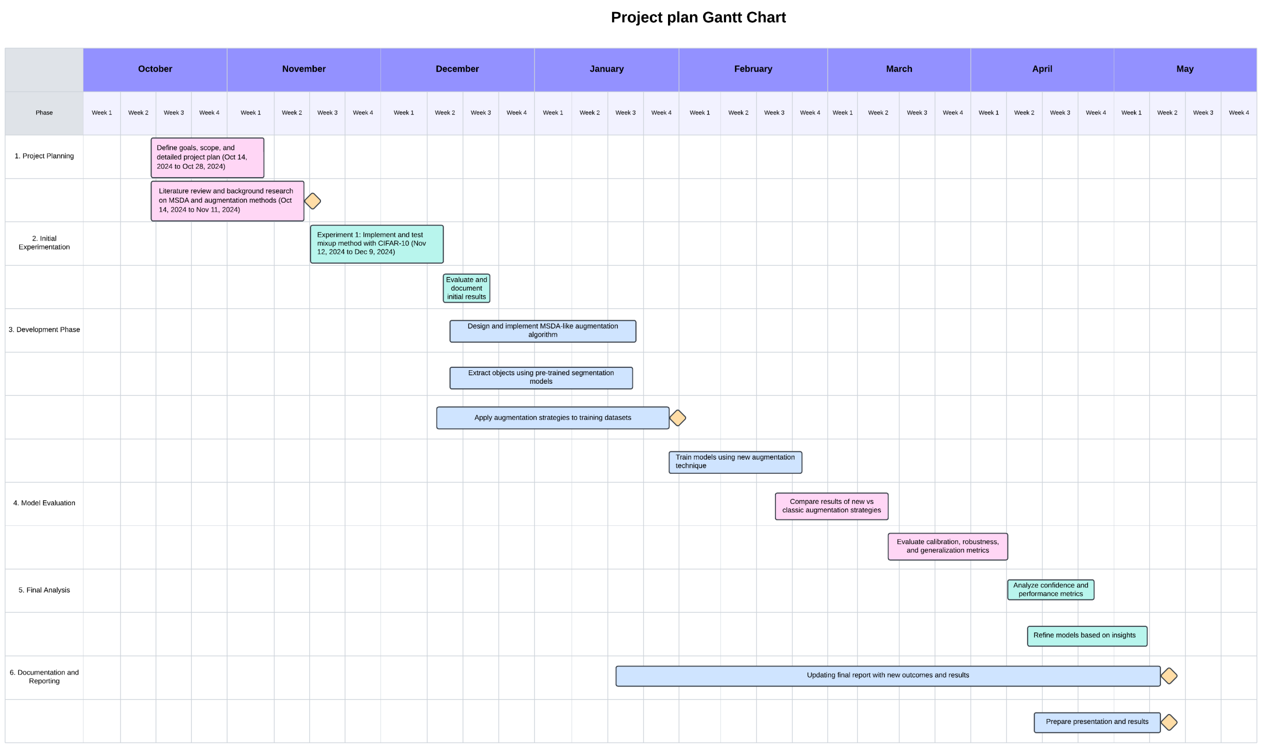

Figure 15: Gantt Chart for my proposed project timeline

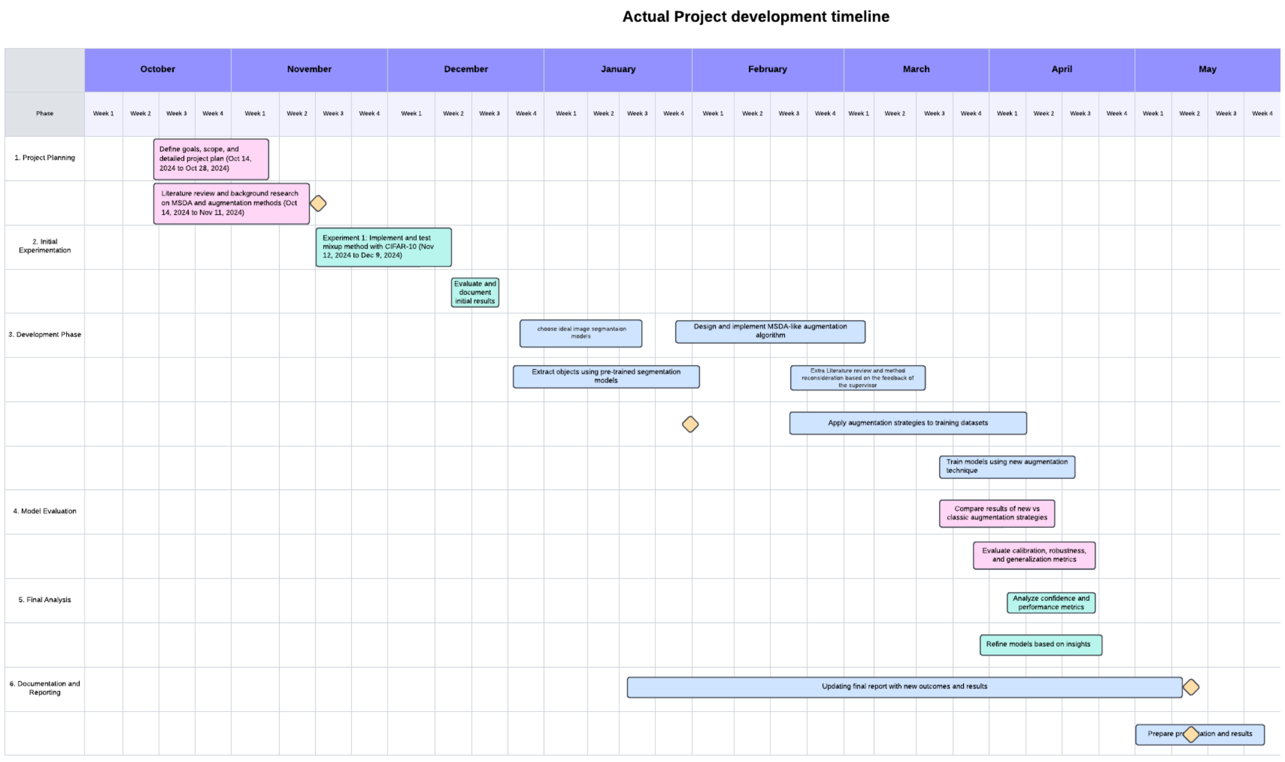

Figure 16: Gantt Chart for my actual project timeline

10 Appendix

A PASCAL VOC Dataset Details

The PASCAL VOC 2012[15] dataset is used for semantic segmentation and object detection. It contains 20 object classes, listed below:

- Aeroplane
- Bicycle
- Bird
- Boat
- Bottle
- Bus
- Car
- Cat
- Chair
- Cow
- Dining table
- Dog
- Horse
- Motorbike
- Person
- Potted plant
- Sheep
- Sofa
- Train
- TV monitor

The dataset includes 1,464 training images and 1,449 validation images with pixel-level annotations. For more details, see [15].

B Model Architecture

The DeepLabV3 model with a MobileNetV3 backbone was used. The architecture modifications include:
The model was pre-trained on a COCO subset with PASCAL VOC labels.

Table 9: Modified MobileNetV3 Architecture

Layer	Output Channels	Notes
Conv1	64	3x3, stride=1, padding=1
MaxPool	-	Identity (disabled)
FC	200	Adapted for 200 classes

C Training parameters

Table 10: Training Configuration for different experiments, using SGD with Momentum optimizer

Dataset	**Model**	**Epochs**	**Scheduler (milestones)**	**LR**	**Batch Size**
CIFAR-10	ResNet-18	150	50, 100	0.1	128
CIFAR-100	ResNet-18	150	50, 100	0.1	128
TinyImageNet	ResNet-18	200	100, 150	0.1	128

D Machine Learning Environment

This appendix lists the versions of Python and machine learning-related libraries used in the project. The environment was set up on a Linux system.

Table 11: Python and Machine Learning Library Versions

Component	**Version**
Python	3.10.14
keras	3.9.2
matplotlib	3.10.0
numpy	1.26.4
nvidia-cuda-cupti-cu12	12.4.127
pandas	2.2.3
scikit-image	0.25.2
scikit-learn	1.6.1
scipy	1.15.2
tensorboard	2.19.0
tf_keras	2.19.0
torch	2.5.1
torchbearer	0.5.5
torchvision	0.20.1

E Project Brief

Enhancing Model Performance: The Impact of Data Augmentation Techniques in machine learning

Yiwei Liao
yl38u22@soton.ac.uk
Supervised by Prof. Jonathan Hare

October 2024

1 Project Description

1:Problem: Data augmentation plays a significant role in machine learning. Machine learning models are extremely sensitive on the training data provided.A well-formed augmentation can enhance the learning process and increase model accuracy and confidence.While traditional augmentation methods, such as mix-up, have had a major impact on model performance, more research is needed to understand how diverse strategies affect accuracy, confidence, and calibration. Current approaches frequently use relatively simple transformations. transformations. However, the impact of more advanced strategies, such as object placement from other datasets or automatic extraction through segmentation algorithms, has yet to be fully explored.

2:Goals: This project will mainly focus on 1:To create an MSDA-like augmentation algorithm that automatically places objects on training images. Objects can be recovered from pre-trained segmentation models or external datasets with existing segmentations. 2:Train models using this new augmentation strategy and compare the outcomes to models trained using classic MSDA methods, such as mix-up. 3: Evaluate model performance in terms of accuracy, robustness, and confidence calibration, with a focus on generalization across datasets. 4: Analyze predictions at various confidence levels and determine the confidence calibration of models trained with new and classic augmentations. 5: Investigate and assess how new augmentation procedures can reduce overconfidence in predictions while potentially improving model resilience and performance metrics.

3:Scope This project will build and compare data augmentation approaches for training convolutional neural network models (e.g., VGG-16, ResNet-18) using datasets like CIFAR-10 or other appropriate ones. The augmentation will perform object placement using segmentation or extraction. Model performance will be evaluated using measures such as accuracy, standard deviation, and proportion of wrong predictions at different confidence levels. The PyTorch framework will be used for training, and model seeds will be modified to achieve strong and consistent results. The new augmentation technique will be compared to established MSDA approaches to determine their impact on model generalization and calibration.

References

[1] Chanwoo Park, Sangdoo Yun, and Sanghyuk Chun. A unified analysis of mixed sample data augmentation: A loss function perspective, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/e6f32e64b9c27d153b46c94f0fe22b56-Paper-Conference.pdf.

[2] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv:1710.09412 [cs, stat]*, 04 2018. URL https://arxiv.org/abs/1710.09412.

[3] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019. URL https://arxiv.org/abs/1905.04899v2.

[4] Ethan Harris, Antonia Marcu, Matthew Painter, Mahesan Niranjan, Adam Prugel-Bennett, and Jonathon Hare. Fmix: Enhancing mixed sample data augmentation. *arXiv (Cornell University)*, 05 2021.

[5] Huafeng Qin, Xin Jin, Hongyu Zhu, Hongchao Liao, Mounîm A El-Yacoubi, and Xinbo Gao. Sumix: Mixup with semantic and uncertain information. *Lecture notes in computer science*, pages 70–88, 11 2024. doi: 10.1007/978-3-031-73021-4_5.

[6] Özgür Özdemir and Elena Battini Sönmez. Attention mechanism and mixup data augmentation for classification of covid-19 computed tomography images. *Journal of King Saud University - Computer and Information Sciences*, 07 2021. doi: 10.1016/j.jksuci.2021.07.005.

[7] Jinyu Zhao, Yichen Zhang, Xuehai He, and Pengtao Xie. Covid-ct-dataset: A ct scan dataset about covid-19. *arXiv:2003.13865 [cs, eess, stat]*, 03 2020. URL https://arxiv.org/abs/2003.13865.

[8] Terrance DeVries and Graham W. Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv:1708.04552 [cs]*, 11 2017. URL https://arxiv.org/abs/1708.04552.

[9] Ahmad Luthfi Azmi Haikal, Novanto Yudistira, and Achmad Ridok. Comprehensive mixed-based data augmentation for detection of rice leaf disease in the wild. *Crop Protection*, 184:106816, 06 2024. doi: 10.1016/j.cropro.2024.106816. URL https://www.sciencedirect.com/science/article/abs/pii/S0261219424002448?via%3Dihub.

[10] Alex Krizhevsky. Cifar-10 and cifar-100 datasets, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

[11] Imagenet. URL https://image-net.org/.

[12] ecs vlc. Github - ecs-vlc/fmix: Official implementation of 'fmix: Enhancing mixed sample data augmentation', 2020. URL https://github.com/ecs-vlc/FMix.

[13] Linda G Shapiro and George C Stockman. *Computer Vision.* Pearson, 2001.

[14] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arxiv.org*, 06 2017. URL https://arxiv.org/abs/1706.05587.

[15] The pascal visual object classes homepage, 2012. URL http://host.robots.ox.ac.uk/pascal/VOC/.

[16] Cityscapes dataset – semantic understanding of urban street scenes. URL https://www.cityscapes-dataset.com/.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, Zitnick C Lawrence, and Piotr Dollár. Microsoft coco: Common objects in context, 2014. URL https://arxiv.org/abs/1405.0312.

[18] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. *2017 IEEE International Conference on Computer Vision (ICCV)*, 10 2017. doi: 10.1109/iccv.2017.322.

[19] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation, 2014. URL https://arxiv.org/abs/1411.4038.

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 05 2015. URL https://arxiv.org/abs/1505.04597.

[21] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network, 04 2017. URL https://arxiv.org/abs/1612.01105.

[22] Multimodal brain tumor segmentation challenge 2020: Data — cbica — perelman school of medicine at the university of pennsylvania. URL https://www.med.upenn.edu/cbica/brats2020/data.html.

[23] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V Le, and Hartwig Adam. Searching for mobilenetv3, 2019. URL https://arxiv.org/abs/1905.02244.

[24] Alex Krizhevsky. Learning multiple layers of features from tiny images, 2009. URL https://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf.

[25] Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge, 2015. URL https://cs231n.stanford.edu/reports/2015/pdfs/yle_project.pdf.

[26] wkentaro. wkentaro/labelme, 11 2019. URL https://github.com/wkentaro/labelme.

[27] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks, 2018. URL https://openaccess.thecvf.com/content_eccv_2018_workshops/w25/html/Wang_ESRGAN_Enhanced_Super-Resolution_Generative_Adversarial_Networks_ECCVW_2018_paper.html.

[28] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv:1605.07146 [cs]*, 06 2017. URL `https://arxiv.org/abs/1605.07146`.

[29] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv:1708.07747 [cs, stat]*, 09 2017. URL `https://arxiv.org/abs/1708.07747`.

[30] Launching the speech commands dataset, 2017. URL `https://research.google/blog/launching-the-speech-commands-dataset/`.

[31] Li Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *2004 Conference on Computer Vision and Pattern Recognition Workshop.* doi: 10.1109/cvpr.2004.383.