October 12, 2022

TECHNICAL UNIVERSITY OF CRETE
School of Electrical and Computer Engineering

Diploma Thesis title:

Data Augmentation Methods for Vision Transformers

Author: Christos Georgakilas

TUC Supervisor: Prof. Michalis Zervakis, Technical University of Crete
Thesis Supervisor: Prof. Nikos Komodakis, University of Crete

Acknowledgments

I am very grateful to Prof. Nikos Komodakis for offering me the opportunity to work on a very interesting research project. I would like to thank him for his continuous guidance throughout the academic year. Many thanks to Profs. Zervakis and Lagoudakis for being two of my teachers at TUC-ECE that provided me with the necessary theoretical background and triggered my interest in the scientific area of the diploma thesis. Additionally, I would like to thank them a lot for evaluating my work.
Completing the 5 years of ECE program, I would like to say that fruitful interactions with all my professors and co-students at TUC were an important part of my study course.

Finally, I would like to thank my family as well as all my friends and my girlfriend who were always there for me when I needed them. Special thanks to my brother for his continuous support and guidance in personal and academic matters.

Abstract

The Transformer architecture was first introduced in 2017 and has since become the standard for Natural Language Processing tasks, replacing Recurrent Neural Networks. For the first time, in 2021, the Transformer architecture was used with great success for computer vision tasks, proving that a Vision Transformer can, under certain conditions, outperform Convolutional Neural Networks and become the state-of-the-art in image recognition. One of the main challenges being tackled by subsequent work on Vision Transformers is the need of the architecture for humongous amounts of data during pre-training in order to achieve state-of-the-art accuracy on the downstream task. Some works have addressed this by altering or adding parts to the original Vision Transformer architecture while others are using Self-Supervised Learning techniques to take advantage of unlabeled data. This thesis explores data augmentation methods for Vision Transformers with the goal to increase the model's accuracy and robustness on classification tasks, with limited amounts of data. Our augmentation methods are based on the architecture's characteristics such as the self-attention mechanism and the input of discrete tokens. All methods are tested for the benchmark classification datasets CIFAR-10 and CIFAR-100 using Supervised Learning and yield great results. When training with the same model hyperparameters, our best augmentation method improves the baseline's accuracy on CIFAR-10 and CIFAR-100 by 1.98 % and 2.71 % respectively.

Table of Contents

Computer Vision

Computer vision is a subfield of Artificial Intelligence that allows computers to perform image recognition. Computer vision algorithms learn from available data collected from vision sensors and then use their knowledge to recognize and react to new unseen image data.

Computer vision is already a crucial factor in today's technology and it will surely be even more prominent in the future as many extremely useful technologies used in real life are based on it. More specifically, computer vision finds applications and shows continuous improvement in multiple sectors such as medical applications, automotive industry, security, entertainment etc..

Computer vision applications:

The potential of computer vision in the medical sector lies particularly in medical image analysis and robotic surgery. Computer vision models have shown to be able to predict and classify specific medical conditions with great precision based on data on which they have been trained on. The use of this technology can alert the doctors to look deeper into a specific case and make a better verdict for the patient's treatment. Also, this technology could further improve with the remote testing of patients. Robotic surgery under the supervision of a doctor is already in use and if correctly developed in the future can be very crucial, as a machine can have greater move precision than a human, a feature very important when it comes to surgery. Obviously, a robotic surgery system of the future should have a perception of its own when conducting surgery.

Autonomous vehicles are also a very promising technology of the future based mainly on computer vision for perception. The system of the car should be able to process the data read by the sensors, and understand the contents of it. Then, it should be able to take action in order to keep the driver and everyone else on the road safe. Cutting-edge image segmentation techniques are used widely already and are being improved, so that the car can understand the data and make correct actions. Though fully autonomous vehicle technology is still not ready to be deployed, companies working on autonomous vehicles have reported that the partial autonomy of the car can already improve safety on the roads.

In entertainment, scene reconstruction, image and scene generation techniques and generally 3D computer vision can greatly impact the creation of a photorealistic digital world.

As for safety, computer vision systems can be used for identification and against possible fraud that could be caused from the generation of 3D fake content (Deepfakes).

Without a doubt, computer vision is already essential in technology and will keep developing further and probably making our life easier in the years to come.

Image 1: Image segmentation essential for autonomous driving. Image from [27]

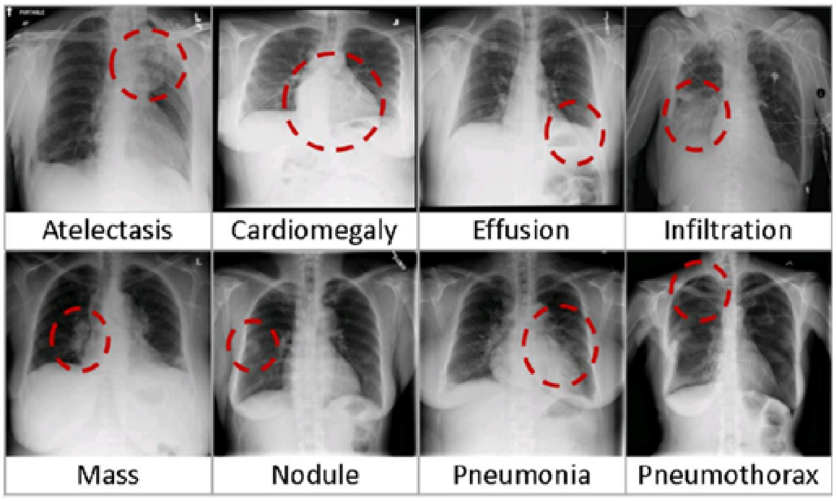


Image 2: Image classification essential for medical image analysis. Image from [28]

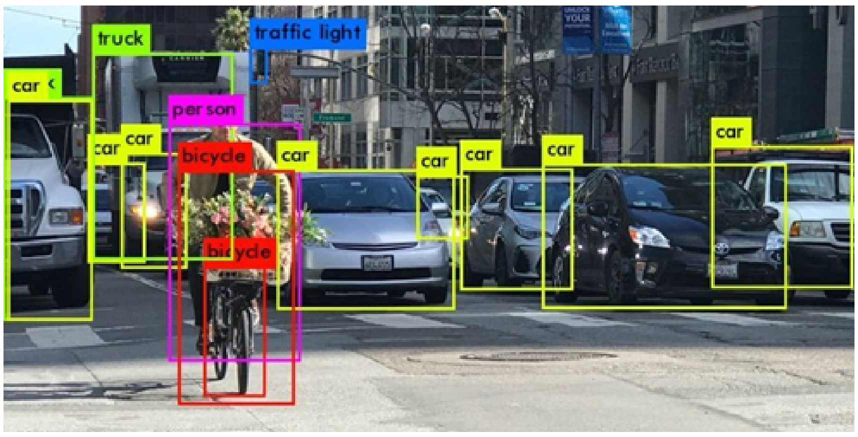


Image 3: Object detection essential for autonomous driving . Image from [29]

Computer vision algorithms:

In the past, computer vision systems were based on manually engineered feature extraction combined with traditional machine learning methods for image recognition (e.g extracted features fed in a Support Vector Machine (SVM) for classification).

In the last couple of years, deep learning has revolutionized the field of computer vision as systems that use deep learning algorithms extract and learn the most important features of the image data by themselves. These new algorithms have significantly improved the models' accuracy compared to more traditional approaches.

The most well-established deep learning algorithm for image recognition tasks is the convolutional neural network (CNN). In recent years, the CNN has undoubtedly been the state-of-the-art algorithm for computer vision due to its great accuracy combined with computational efficiency. In 2021, a new deep learning algorithm for image recognition was developed, called the Vision Transformer (ViT). The ViT lacked the data-efficiency of the CNN but demonstrated excellent performance when there was an abundance of data. Since then, the ViT architecture has been used widely and its variations have outperformed CNNs in multiple image recognition benchmarks.

Finally, for computer vision to be truly evolved in future technologies, research on deep learning models should focus on making them more reliable, secure, data-efficient and as generalizable as possible among different recognition tasks. The work of this thesis mainly focuses on improving the data-efficiency of the ViT.

1. Theoretical Background

1.1 Transformer Architecture

The Transformer was originally introduced in 2017 [1] as a simple neural network encoder-decoder architecture based purely on the attention mechanism to tackle sequence modeling problems such as translation of text streams. Up until then, sequence to sequence state-of-the-art models were mostly based on Recurrent or Convolutional neural networks (CNNs) [2, 3]. The Transformer has since become the state-of-the-art for Natural Language Processing tasks regarding accuracy and computation as a result of parallelization.

The encoder of the transformer is made up of alternating layers of multi-head self-attention and position-wise fully connected feed-forward networks as described in [1]. Residual connections [4] and layer normalization [5] are also applied between layers.

The decoder has a very similar architecture to the encoder, but between the multi-head attention and position-wise fully connected feed-forward networks, a layer of multi-head attention is applied on the output of the encoder.

The input of the transformer (to the encoder) is a sequence of discrete tokens $(x_1, x_2, \dots, x_n)$, e.g words in a sentence, and the output (from the decoder) is another -possibly different-sized- sequence of discrete tokens$(y_1, y_2, \dots, y_n)$.

1.2 Vision Transformer

The Vision Transformer (ViT) was established in 2021 [6] when for the first time, a Transformer- like architecture without the use of convolutions, achieved state-of-the-art performance on image recognition tasks.

1.2.1 Architecture

The main block of the original ViT architecture [6] is almost identical to the encoder of the Transformer [1]. More specifically, the core of the architecture consists of alternating layers of multi-head self-attention and multilayer perceptrons (MLP). Layer normalization is applied before each layer and residual connections are applied after it (see Figure 2). The input of the architecture is a sequence of all the patches of the image. The patches are non-overlapping and of fixed-size. For a classification task, the final output is the predicted class of the image.

The functionality of the model based on [6] is described below and visualized in Figure 1:

- The image x is split into $N = \frac{H \cdot W}{P^2}$ non-overlapping fixed-size patches $(x_1, x_2, \dots, x_N)$, where H is the height and W is the width of the original image and (P, P) is the resolution of each patch, with $P^2 = H_i \cdot W_i$.

- The Transformer encoder architecture requires the input to be a sequence of 1D tokens, so as described in [6] each 2D image patch is reshaped from being $x_i \in R^{H_i\, x\, W_i\, x\, C}$ to $x_{i'} \in R^{(P^2 \cdot C)}$ where H_i is the height of each patch, W_i is the width of each patch and C is the number of color channels. The input sequence $(x_{1'}, x_{2'}, \dots, x_{N'})$ of 1D tokens for the Transformer has a length of N as described in the first step.

- A trainable linear projection E maps all patches to D dimensions - which is the constant latent vector size of the Transformer - and outputs the patch embeddings. The patch embeddings $(z_1, z_2, \dots, z_N) \in R^D$ as $z_i = x_{i'} \cdot E$, where $E \in R^{(P^2 \cdot C)\, x\, D}$.

- A learnable token embedding $z_0 \in R^D$ called the class token is concatenated at the beginning of the sequence of patch embeddings for each image. The state of this token changes during the forward pass and at the output of the encoder it contains the image representation. It is then fed in a classification head which outputs the probability of the image to belong in each class. As mentioned in a later work [7] of the authors, the replacement of the class token by global average-pooling (GAP) [8, 9] might be beneficial.

- Unlike CNNs, the ViT takes as input the patch embedding sequence that on its own contains no 2D spatial information of the image. To give an order context to the sequence, which is also very important for the attention mechanism, positional embeddings are added to all patch embeddings. The resulting sequence of size $(N + 1)\, x\, D$ is the input of the transformer encoder.

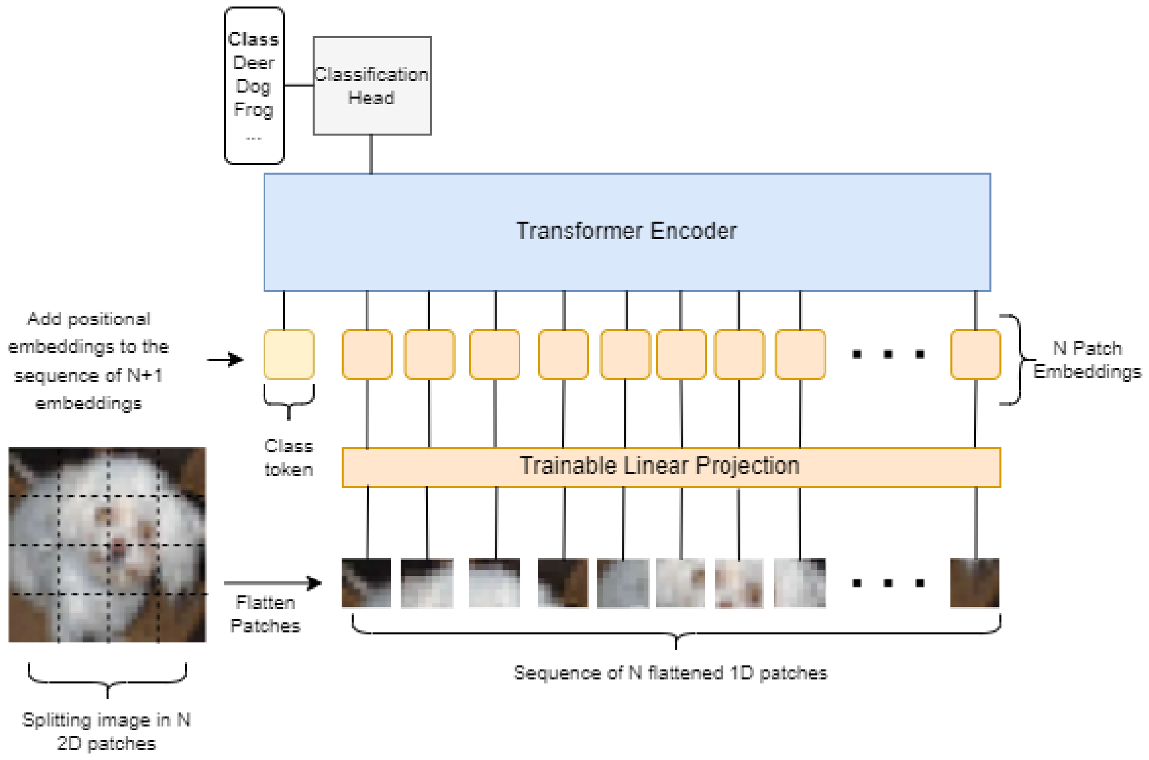

Class
Deer
Dog
Frog
...
Classification Head
Transformer Encoder
Add positional embeddings to the sequence of N+1 embeddings
N Patch Embeddings
Class token
Trainable Linear Projection
Flatten Patches
Sequence of N flattened 1D patches
Splitting image in N 2D patches

Figure 1: Vision Transformer model functionality. The functionality as well as the figure itself are based on [6]. The image shown belongs in the CIFAR-10 test batch and is used here purely for demonstration purposes.

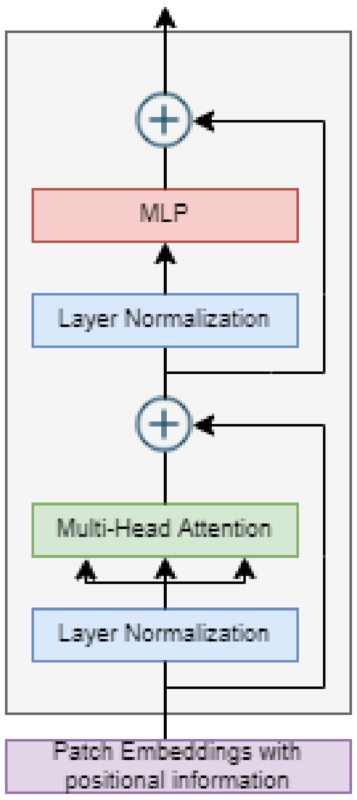


Figure 2: Detailed representation of the layers that compose the transformer encoder block based on [6]. Consecutive transformer encoder blocks are stacked to implement the ViT.

1.2.2 Positional Embeddings

Adding some form of positional information to the patch embeddings before feeding them to the transformer encoder is deemed necessary to improve accuracy of the ViT [6]. In this original work, several positional embedding approaches are tested and ultimately the one being used is the learnable 1D positional embedding. According to the authors, the 1D learnable embeddings learn the 2D image topology sufficiently and thus the usage of more sophisticated 2D positional embeddings is not necessary. In a later work [7], the authors switch to fixed 2D sin-cos positional embeddings [8] and observe great improvement in performance.

1.2.3 Self-Attention

In CNNs, through the convolutional layers, a new representation of each image pixel is calculated by summing all the pixel values in the local neighborhood and weighting them by the values of the kernel (dot product).

In the ViT where the input is a sequence of tokens $z \in R^{N+1\,x\,D}$, the self-attention mechanism allows the tokens of the image to interact with each other on a global scale. Through this interaction in the attention layer, a new representation of each token is computed.

The operation for each token is the same and starts by calculating the attention scores between the individual token, itself and all other tokens in the sequence. Then, the sum of all the token representations (v) weighted by the attention scores already calculated, results in the new representation of the individual token (Eq. 3). The attention scores A_{ij} are derived by calculating the pairwise similarity between the individual query (q_i) and key (k_j) representations of two tokens of the sequence (Eq. 2).

Equations by [6, 1]:

$[q, k, v] = z \cdot U_{qkv}$, calculation of q, k, v parameters, $U_{qkv} \in R^{D\,x\,3D_h}$ **(Eq. 1)**

$A_{ij} = softmax\left(\frac{qk^T}{\sqrt{D_h}}\right)$, calculation of A_{ij} attention scores $A_{ij} \in R^{(N+1)\,x\,(N+1)}$ **(Eq. 2)**

$SA(z) = A_{ij} \cdot v$, calculation of the new token representations **(Eq. 3)**

1.2.4 Multi-Head Self-Attention

In practice, the ViT architecture contains layers of multi-head self-attention (MSA). Multi-head self-attention is the application of k self-attention operations in parallel, followed by the projection of their concatenated outputs (Eq. 4).

Each operation is called an attention "head" and the number of heads is a model hyperparameter. Each attention head calculates different attention scores between tokens and thus attends the sequence differently. This allows the model to compute richer representations for the input data leading to overall better performance.

Calculating Multi-head attention of token sequence $z \in R^{N+1\, x\, D}$, where $D_h = \frac{D}{k}$ to keep compute and number of parameters constant when changing number of heads (k).

Equation by [6]:

$$MSA(z) = [SA_1(z); SA_2(z);...; SA_k(z)]\, U_{MSA} \qquad , U_{MSA} \in R^{k \cdot D_h\, x\, D} \qquad \textbf{(Eq. 4)}$$

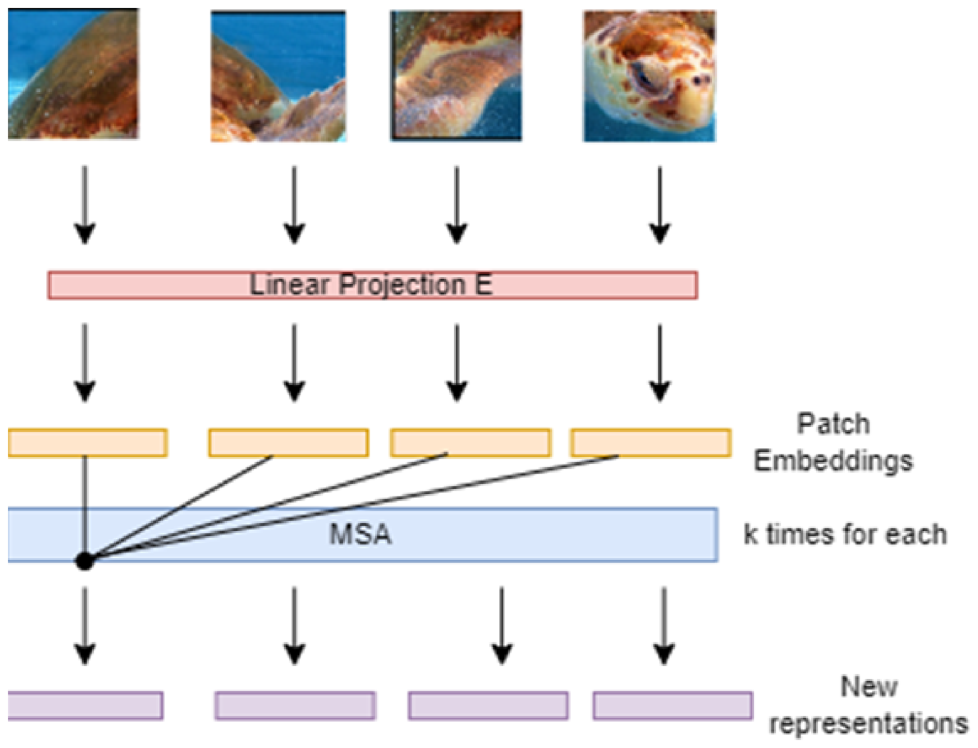


Image 4: MSA operation visualization. Turtle image taken from test set of Imagenet-100 dataset.

1.2.5 Drawbacks

The main drawback of the ViT architecture is its need for excessive amounts of data during pre-training compared to CNNs in order to achieve state-of-the-art accuracy in downstream tasks. This issue is addressed already from the initial ViT paper [6] and is mainly attributed to the architecture's lack of inductive bias.

The inductive bias of a CNN such as 2D neighborhood structure, locality and the translation equivariance as a result of weight sharing allow the model to perform induction from a smaller amount of training data compared to the ViT.

In ViTs inductive bias is lacking, as only MLP layers are local and translationally equivariant and the 2D neighborhood structure is only used in the beginning when neighboring pixels of the image are generally placed in the same patch. An inductive bias about the 2D spatial relations of different patches can also be introduced if fixed positional embeddings are used for training. In the case of learnable positional embeddings, no such bias exists.

As a result, to achieve state-of-the-art, strong data augmentation techniques are necessary when training the ViT with limited amounts of data. On the other hand, the lack of inductive bias can be useful for ViTs when very large amounts of data are available as the model is more flexible to learn from the data and thus can generalize better [6].

1.3 Data Augmentation

Data Augmentation is an empirically effective technique, used when training deep learning models. As its name suggests, data augmentation is the process of augmenting the available training dataset by creating new data samples based on existing ones. The goal of data augmentation is to help the model avoid overfitting and thus better generalize to unseen samples (test dataset).

Though data augmentation techniques vary, the common objective of all the techniques is to create unique data samples with useful information while also increasing the variance of the dataset. Of course, the type and the amount of data augmentation differ depending on the type of data, the deep learning model, the dataset etc.. The aforementioned parameters of data augmentation are commonly determined empirically and experimentally, but there is also work that seeks to theoretically analyze and determine them [10].

Common augmentation techniques used for image datasets are, among others, horizontal flip, vertical flip, rotation, scaling, blurring etc..

Data augmentation techniques are particularly necessary when the available training data are limited and the model in hand requires large amounts of data.
Accordingly, this thesis studies how specific data augmentation techniques can help the ViT model be more robust and accurate in a data-efficient manner.

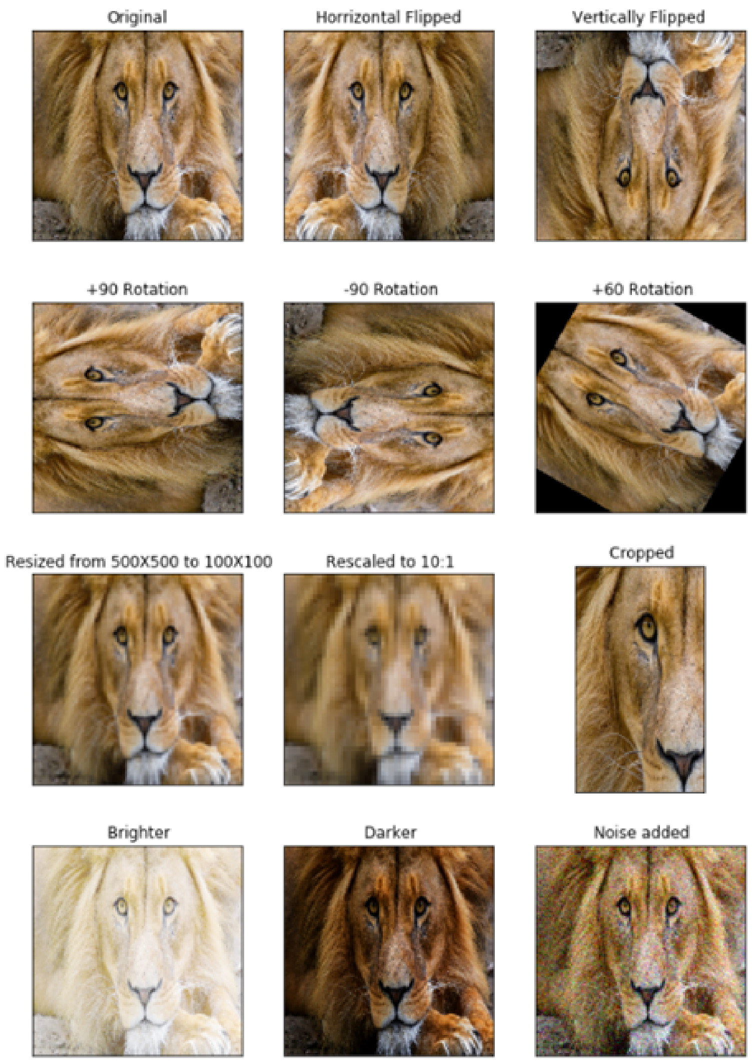


Image 5: Data augmentations . Image from [30]

2. Introduction

In recent years, CNNs [11] have been the standard deep learning architecture used across most vision tasks, achieving state-of-the-art (SOTA) performance. In 2021, the ViT [6], an architecture using the attention mechanism and without any convolutions surpassed its CNN counterparts, achieving SOTA in various recognition benchmarks. Since then, many succeeding works have used variations of the ViT for different vision tasks with great success.

The main disadvantage of the ViT, expressed already from the original work [6] is the model's need for immense amounts of data during pre-training in order to consistently outperform the SOTA CNN architectures in downstream tasks. This need for extra data compared to CNNs is ascribed to ViT's lack of inductive bias.

The original work and subsequent research have increasingly addressed this data-efficiency problem in multiple ways, through architectural variations, self-supervision and strong data augmentations.

2.1 Architecture-Based Solutions

The original ViT work introduced a model which combines CNNs and Transformers, named the "Hybrid Architecture". In this architecture, the trainable linear projection E is applied on an input created from feature maps of a CNN, instead of being applied on raw image patches as in the original ViT. The goal of this is to introduce some inductive bias through CNNs in the early stages of the model and possibly create more informative input representations, which can be very helpful when data is limited. This model is experimentally tested and is proven to indeed outperform the ViT in smaller datasets and smaller computational budgets [6]. For larger models and larger datasets, the plain ViT outperforms this model. Further research has yet been conducted on the combination of the ViT with Convolutional layers to improve performance and data-efficiency [12].

Another really important follow-up work [13] addresses the problem through a small architectural modification as well as strong data augmentations. This work introduces a transformer-specific knowledge distillation operation where a teacher network teaches a student network through attention. The student network is a ViT while the teacher network can either be another ViT or a CNN, though through experimentation, it is observed that having a CNN as the teacher network is beneficial for the system's performance. The distillation operation presented requires a distillation token which is used complementary to the class token and seeks to reproduce the image label predicted by the teacher network. Apart from that, the distillation token is part of the input and output token sequence of the student ViT and thus interacts with all other tokens through self-attention. This allows the model to learn from both the output of the teacher network and the true label predicted by the class token. Combining this method with general pre-existing data augmentation techniques outperforms plain ViT and competes with SOTA models on Imagenet [14] without external data. The authors suggest that transformer-specific data augmentation techniques can be very beneficial tools to further improve the model's performance.

2.2 Data-Based Solutions

The data-based solutions to the data-efficiency problem mainly utilize:

1) Self-Supervised Learning, in order to use unlabelled data during the pretext task and find useful data patterns from it for the downstream task

2) Data augmentation techniques, in order to augment the dataset, providing more, diverse data samples for pre-training

2.2.1 Self-Supervised Learning

A very significant work on the self-supervised pre-training approach for ViTs is based on masked autoencoders [15]. The autoencoder model proposed is composed of an asymmetric encoder-decoder architecture which results in significant computational efficiency. The encoder of the architecture is a ViT and is used both during pre-training and for the downstream recognition task.

The pretext task is based on masked image modelling. More specifically, to create a challenging task, a large part of the input image is masked before being fed into the encoder. The encoder maps the image input to a latent representation from which the decoder seeks to reconstruct the original image.

The main differences between this approach and previous autoencoders is the asymmetric encoder-decoder design as well as the fact that masked patches are "dropped" instead of being fed to the encoder. The latter becomes much simpler to implement in ViTs compared to CNNs due to the usage of discrete image patches. The large encoder of the MAE is applied only on the unmasked patches (25% of the image) while the lightweight decoder reconstructs from the latent representation and the masked patches. This not only allows the encoder to be larger without a significant computational cost, but through experimentation it is also shown that the encoder is benefitted in accuracy from only seeing real image patches and not masked ones. When fine-tuning the encoder on ImageNet, the model outperformed all results that used Imagenet without external data at the time, indicating the great potential of Self-Supervised approaches for efficient ViT training.

2.2.2 Data Augmentation Techniques

Data augmentation techniques are used across all recent research on the ViT as their ability to increase performance has been proven through thorough testing [13, 7]. In [13], the data augmentations used are random erasing [16], RandAugment [17], Mixup [18] and Cutmix [19] . Random erasing is a data augmentation technique, originally used to train CNNs, which randomly masks a rectangular region of the image by replacing its original values with random ones. Samples with different occlusions are generated which makes the model more robust to occlusion and helps it avoid overfitting. RandAugment is composed of multiple augmentation techniques such as rotation, x,y - translation, ColorJitter etc.. In [7], a small amount of RandAugment and Mixup augmentations are also used and prove to be beneficial, while further augmentations such as Cutmix are considered.

Various image mixing augmentations such as the ones mentioned above [18, 19] are becoming increasingly prevalent in training deep learning models.

The first image mixing augmentation introduced, Mixup [18], creates new data samples by linearly combining pairs of existing ones. The two data samples being mixed are drawn randomly from the training set and their weights $\lambda \, and \, (1-\lambda) \in [0,1]$ in the linear combination are also random. The new data sample's ground truth label is also determined by the linear combination of the pair's labels with the respective weights. Though the resulting mixed images look unnatural, the Mixup method generally improves classification performance and increases robustness to adversarial examples.

The Cutmix [19] augmentation, especially useful for training CNNs, was subsequently introduced as an image mixing technique. Though it shares some similarities with Mixup, Cutmix also utilizes regional dropout [16, 20] which is proven to be effective for training CNN classifiers. Regional dropout augmentations randomly mask a rectangular region of the image by replacing its original values with random ones or zeros. This way, the model attends to more parts of the image during training and not only the most distinct ones, improving generalization, robustness and classification performance. The authors of Cutmix identify the loss of useful information due to regional dropout as a problem and seek to maximize the information fed in the network while also keeping the generalization improvements from dropout. To do this, Cutmix augmentation replaces the region of the original image removed during dropout with an equally-sized patch from another image. The ground truth label of the new data sample is a proportional mix of the pair's labels based on the number of pixels that each image contributes. This approach maintains and improves the generalization and robustness benefits of regional dropout while simultaneously ensuring that all pixels in the training images are informative.

Follow-up augmentation methods for CNNs based on Cutmix emerged [21, 22], which argue that randomly mixing a patch of a source image to a target image is probably not optimal as the same label is assigned to the new data sample regardless of the final information contained in it. This label noise on the new data samples can cause instability during training. Consequently, these augmentation techniques suggest that it is more beneficial to mix image patches which are the most salient when determining their source image's label. By doing so, the label mixing of the new data sample better reflects the actual percentage of each image being mixed. These saliency mix methods have shown to improve model robustness and outperform Cutmix when training SOTA CNN models for classification.

Unfortunately, these saliency mixing methods [21, 22] haven't yet been utilized for training ViTs as Cutmix and Mixup are mainly the ones used by recent works.

In this thesis, multiple ViT-specific data augmentation methods are developed and thoroughly examined. All methods proposed utilize the architecture's characteristics such as the self-attention mechanism and the input of discrete tokens. Our main goal is to formulate a saliency mixing augmentation method that is particularly useful for training ViTs. The methods presented in this work are all operating at token level and are generally composed of two steps (two forward passes and one backward pass), the first one being patch dropping and the second one being a combination of additional patch dropping and patch mixing.

The initial patch dropping is done randomly similarly to regional dropout but on a token level. The benefit of the ViT in this regard is that due to the input being a sequence of discrete patches, a masked patch doesn't need to be fed in the model and can simply be dropped out of the sequence. This can be very useful as it reduces computational cost, it avoids showing the model masked patches with useless information and it creates new, different data samples which are used in the next step. Doing this step and dropping some random patches before deciding which patches are the most salient ones proves advantageous for training, as a greater variety of new data samples is created in the mixing step.

In the second step, the attention mechanism of the ViT as well as the class token are used to determine which patches of the effective sequences are the most salient ones. We argue that the most salient patches of an image are the ones that after the last self-attention layer of the ViT have the greatest attention scores with respect to the class token as they contributed the most in its final representation. Depending on the method, additional patches are dropped and some of the patches are mixed between image pairs of different classes. We observe that removing additional patches from the sequence decreases computation, doesn't deteriorate the model and in some cases it might even increase its performance.

Extensive experiments and comparisons are presented for all the data augmentation methods tested. When training the plain ViT with the same model hyperparameters, our best augmentation method improves the baseline's classification accuracy on CIFAR-10 and CIFAR-100 [23] by 1.98 % and 2.71 % respectively.

3. Related Work

Our augmentation techniques are based on the concepts of patch masking and patch mixing, so related work on these 2 topics is listed below.

3.1 Patch Masking

Existing patch masking augmentations [24, 16, 20] have been developed primarily for CNN architectures. As such, Cutout [20] and Random-erasing [16] randomly mask an entire rectangular region of the image by replacing its original values with random ones. This operation creates samples with different occlusions, making the model more robust. Additionally, randomly masking distinctive regions of the image allows the model to attend to more parts of the image during training, avoiding overfitting and improving generalization and classification performance. The size of the regional dropout is constant for Cutout while Random-erasing applies a different-sized (bounded by some values) mask in each training iteration. Hide-and-seek augmentation [24] uses a patch masking technique that is more similar to the patch masking of the ViT. First, the image is being split in a predetermined number of patches. Then, every patch is masked independently with a probability of $p = 0.5$. The resulting image is fed into the model. In this augmentation method, the values of the masked patches are set to zero. This method still differs from token level masking techniques, as the patch size used is 4 times larger than the regular patch size used in the ViT.

3.2 Patch Mixing

Patch mixing augmentation techniques are typically combined with patch masking and further improve the model's robustness and classification performance. The most widely used patch masking augmentation is Cutmix [19] which replaces a random rectangular patch of an image with an equally-sized random patch from another image. The labels of the image pair are proportionally mixed in order to create the label of the new data sample. Saliency based mixing techniques [21, 22] outperform Cutmix by mixing the patches that contain the most information about the image's contents instead of random ones. This allows the labels of the new data samples to be less noisy and thus the training to me more stable. The Attentive CutMix [22] works similarly to patch mixing at token level as the salient regions of the image which are used for mixing are calculated through attention by a pre-trained network and are generally found sparsely in the image. Very recently, mixing augmentation techniques have also been introduced specifically for ViTs [25, 26]. Both of these augmentation techniques randomly drop and mix patches at token level. In TransMix [25], the final ground truth label of the new data sample is computed by using the attention maps from the ViT model itself. In TokenMix [26], the final ground truth label of the new data sample is computed according to the content-based neural activation maps from a pre-trained CNN teacher network. This thesis examines saliency mixing data augmentation techniques that are ViT specific and use attention as the mechanism to detect the salient patches of each image.

4. Methodology

The augmentation methods implemented in this thesis are based on image masking or a combination of image masking and image mixing. In this section of the thesis, only the pipeline of our best augmentation method will be analyzed and visualized thoroughly as it covers the general idea of our approach. The rest of the augmentations will be recorded in detail in the "Experiments" section of the thesis. All data augmentation methods are applied with a probability of 0.5, so the input of the ViT is alternating between batches of only full original images and batches of only augmented images.

Revisiting the functionality of the ViT (Figure 1):

We split the image x into $N = \frac{H \cdot W}{P^2}$ non-overlapping fixed-size patches $(x_1, x_2, \dots, x_N)$, where H is the height and W is the width of the original image and (P, P) is the resolution of each patch, with $P^2 = H_i \cdot W_i$.

Each 2D image patch is then reshaped from being $x_i \in R^{H_i \, x \, W_i \, x \, C}$ to $x_{i'} \in R^{(P^2 \cdot C)}$ where H_i is the height of each patch, W_i is the width of each patch and C is the number of color channels. The input sequence $(x_{1'}, x_{2'}, \dots, x_{N'})$ of 1D tokens for the Transformer has a length of N as described above.

A trainable linear projection E maps all patches to D dimensions - which is the constant latent vector size of the Transformer - and outputs the patch embeddings.
The patch embeddings $(z_1, z_2, \dots, z_N) \in R^D$ as $z_i = x_{i'} \cdot E$, where $E \in R^{(P^2 \cdot C) \, x \, D}$.

A learnable token embedding $z_0 \in R^D$ called the class token is concatenated at the beginning of the sequence of patch embeddings for each image. The state of this token changes during the forward pass and at the output of the encoder it contains the image representation. It is then fed in a classification head which outputs the probability of the image to belong in each class.

Unlike CNNs, the ViT takes as input the patch embedding sequence that on its own contains no 2D spatial information of the image. To give an order context to the sequence, which is also very important for the attention mechanism, positional embeddings are added to all patch embeddings. The resulting sequence of size $(N + 1) \, x \, D$ is the input of the transformer encoder for a single image.

The best augmentation method will be described for an instance of training when a batch of images with length Batch_size is being fed into the ViT. The method is visualized in steps in its subsection and also in its entirety in Figure 3.

4.1 Patch Dropping

First forward pass, no backward pass:

1. The input sequence is of size $Batch\_size \; x \; N \; x \; D$ as the class token hasn't been added yet

2. We add the positional embeddings for all patches and then randomly drop a% of the image patches from each image, so we drop $a * \frac{N}{100}$ patches and are left with a sequence of size $Batch\_size \; x \; (N - a * \frac{N}{100}) \; x \; D$

3. We expand and concatenate the class tokens to the beginning of the sequence and we get a sequence of size $Batch\_size \; x \; (N - a * \frac{N}{100} + 1) \; x \; D$

4. The effective token sequence of size $Batch\_size \; x \; (N - a * \frac{N}{100} + 1) \; x \; D$ is fed as input to the network and we forward pass. The goal of the first forward pass is to calculate the attention scores between all patches of the sequence and their respective class token.

The a% of patches mentioned above are selected to be dropped in the second step based on the following random operation [15]:

After having added the positional embeddings to the sequence and before concatenating the class token, randomly shuffle the list of N patches and remove the last portion of the list, based on the masking ratio specified. This process is equivalent to sampling patches without replacement and the shuffling operation is independent for each image of the batch. This simple implementation introduces negligible overhead as the shuffling operation is very fast. Shuffling the sequence of patches is not a problem for the model as long as positional information has been previously added. The decrease of the effective sequence by a% has an even greater computational benefit for bigger models and datasets with high-resolution images where patch sequences are longer. Other than that, through experiments we observe that dropping a% of patches at random before finding the maximum attention patches of each image used for mixing also improves the overall model performance. This is probably due to the model being forced to find more patterns and attend different parts of the image with greater attention.

Simple random sampling without replacement:

In order to do the aforementioned sampling, we generate a matrix $s \in R^{Batch_{size}\ x\ N}$ with randomly shuffled indices $\in [0, N)$. We then shuffle the patch embeddings of each image in the batch according to s before dropping the last a% of patch embeddings per image.

Since positional embeddings have been added to the sequence before shuffling we don't need to unshuffle the sequence for the ViT to operate on it properly. This makes the operation even more computationally efficient.

Finally, the class token should be added after this sampling and dropping operation as otherwise it might be randomly dropped.

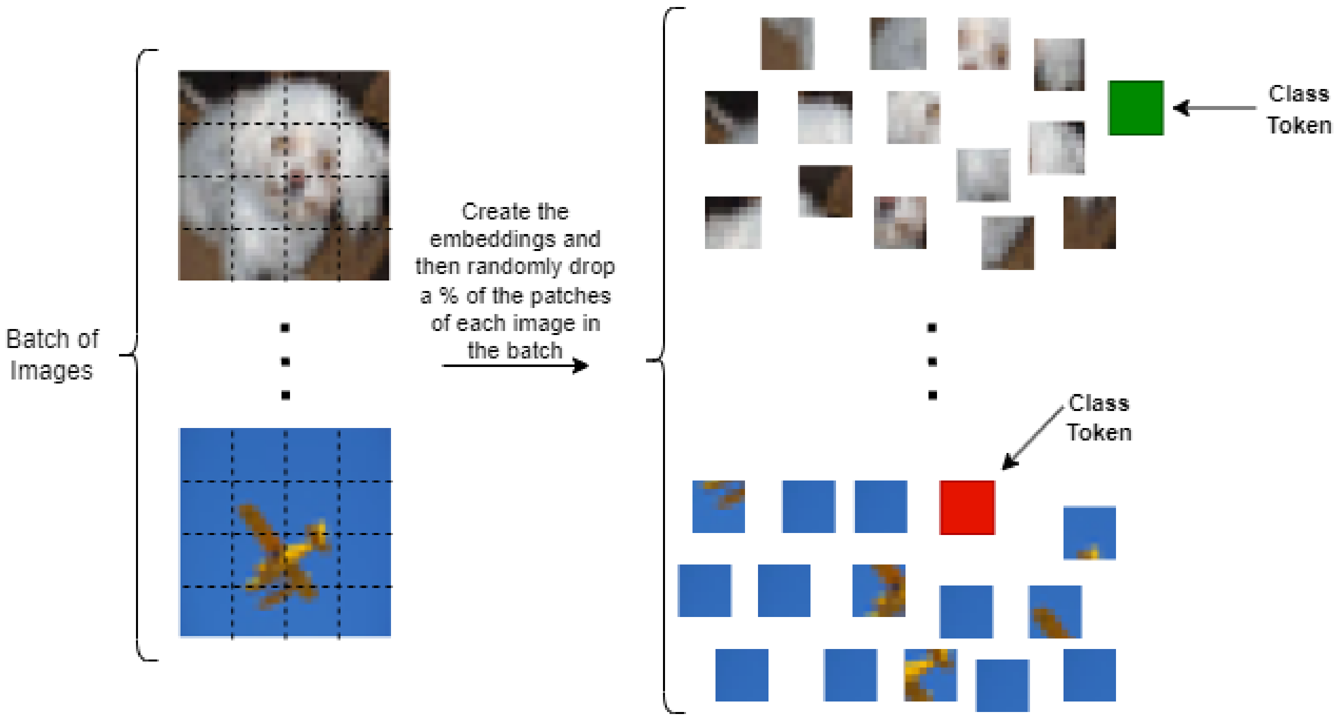

Batch of Images
Create the embeddings and then randomly drop a % of the patches of each image in the batch
Class Token
Class Token

Image 6: Patch dropping visualization. The images shown belong in the CIFAR-10 test batch and are used here purely for demonstration purposes.

4.2 Patch Dropping and Mixing

After doing the first forward pass with the above effective sequence, we calculate the 2b% of patches with the maximum attention for each image of the batch. We then randomly choose half of those patches (b%). In the subsequent (second) forward pass we will drop those b% patches from their original image and inject them to an image of the batch which belongs in a different class.

Second forward pass, then backward pass:

1. The input sequence is the same as before and of size $Batch\_size\ x\ N\ x\ D$ as the class token hasn't been added yet

2. We add the positional embeddings for all patches and then drop the same a% of the image patches as before from each image, so we drop $a * \frac{N}{100}$ patches and are left with the same sequence of size $Batch\_size\ x\ (N - a * \frac{N}{100})\ x\ D$ as before

3. We expand and concatenate the class tokens to the beginning of the sequence and we get the same sequence of size $Batch\_size\ x\ (N - a * \frac{N}{100} + 1)\ x\ D$ as before

4. We form pairs between images of the batch that belong in different classes. The b% of maximum attention patches that is dropped from an image is injected to its pair and vice versa. The patches being injected are concatenated at the end of the target image's sequence. All patches contain the positional information from their original image.

5. Our resulting sequence of size $Batch\_size\ x\ (N - a * \frac{N}{100} + 1)\ x\ D$ contains image samples solely created by the patch dropping and mixing augmentation

6. The effective token sequence of size $Batch\_size\ x\ (N - a * \frac{N}{100} + 1)\ x\ D$ is fed as input to the network and we forward pass. Then, after calculating the appropriate ground truth label for each mixed image of the batch, we backward pass.

The maximum attention patches of the effective sequence are calculated as described below:

After the first forward pass of the model we have computed the attention scores matrix of the last attention layer of the ViT

$$A_{ij} = softmax\left(\frac{qk^T}{\sqrt{D_h}}\right), \quad A_{ij} \in R^{Batch\_size\ x\ Heads\ x\ (N-a*\frac{N}{100}+1)\ x\ (N-a*\frac{N}{100}+1)}$$

where Heads is the total number of attention heads applied in the model.

We want to find the attention scores between each patch of the effective sequence and the Class token which contains the final image representation. The patches with the greatest attention score values with respect to (wrt.) the Class token are the ones that contribute the most to the formation of its representation and thus are the most salient ones when determining the class of the image. These patches are the ones we want to drop from their original image and mix to another image of different class.

To find the indices of those maximum attention patches:

First, we only keep the dimension with the attention scores wrt. the class token as $A_{ij} \in R^{Batch\_size\ x\ Heads\ x\ (N-a*\frac{N}{100}+1)}$.

Then we average across attention heads and get: $A_{ij} \in R^{Batch\_size\ x\ (N-a*\frac{N}{100}+1)}$.

We then keep the indices of the 2b% of patches with maximum attention from this sequence. Finally, through random shuffling we select half of those indices (b%). We use these indices in the second forward pass to drop and mix the patches with maximum attention (We keep 2b% and then randomly pick b% of them in order to provide additional randomness which can help generate more diverse mixed data samples).

In the second step of the second forward pass, we sample the same patches as in the second step of the first forward pass, since we have calculated the indexes of the maximum attention patches based on that particular patch sequence.

Step 4 of the second forward pass (mixing operation):

- We have a sequence of patches after a% drop: $z'_{ijk} \in R^{Batch\_size \; x \; (N - a \cdot \frac{N}{100}) \; x \; D}$
- We also have a matrix $p_{ijk} \in R^{Batch\_size \; x \; (b \cdot \frac{N}{100}) \; x \; D}$ containing the b% max attention patches for all images of the batch
- We drop those patches from their original images and we get the resulting effective sequence $z''_{ijk} \in R^{Batch\_size \; x \; (N - (a+b) \cdot \frac{N}{100}) \; x \; D}$
- We then shuffle p_{ijk} along its 0th dimension so that patches in p_{i**} and z''_{i**} belong in different class $\forall \; i \in [0, Batch_{size})$
- Finally, we concatenate p_{ijk} to the end of z''_{ijk} along the 1st dimension and we have the resulting dropped/mixed images

This image mixing step is the operation with the greatest computational load for the model. With the current implementation, where only images of different classes can be mixed together, substantial constraints have to be introduced. These constraints lead to repetitive accesses of the data that pose a non-negligible load on the model.

Some experiments have been conducted without the constraint of mixing images of different classes and the computational load was less while the accuracy of the model didn’t deteriorate. In these experiments, the matrix p_{ijk} containing the patches to be mixed and dropped from the original image was shuffled randomly and not shuffled according to mixing constraints. More experiments with this method should be tested as it might keep the model’s accuracy the same with non-negligible computational benefits.

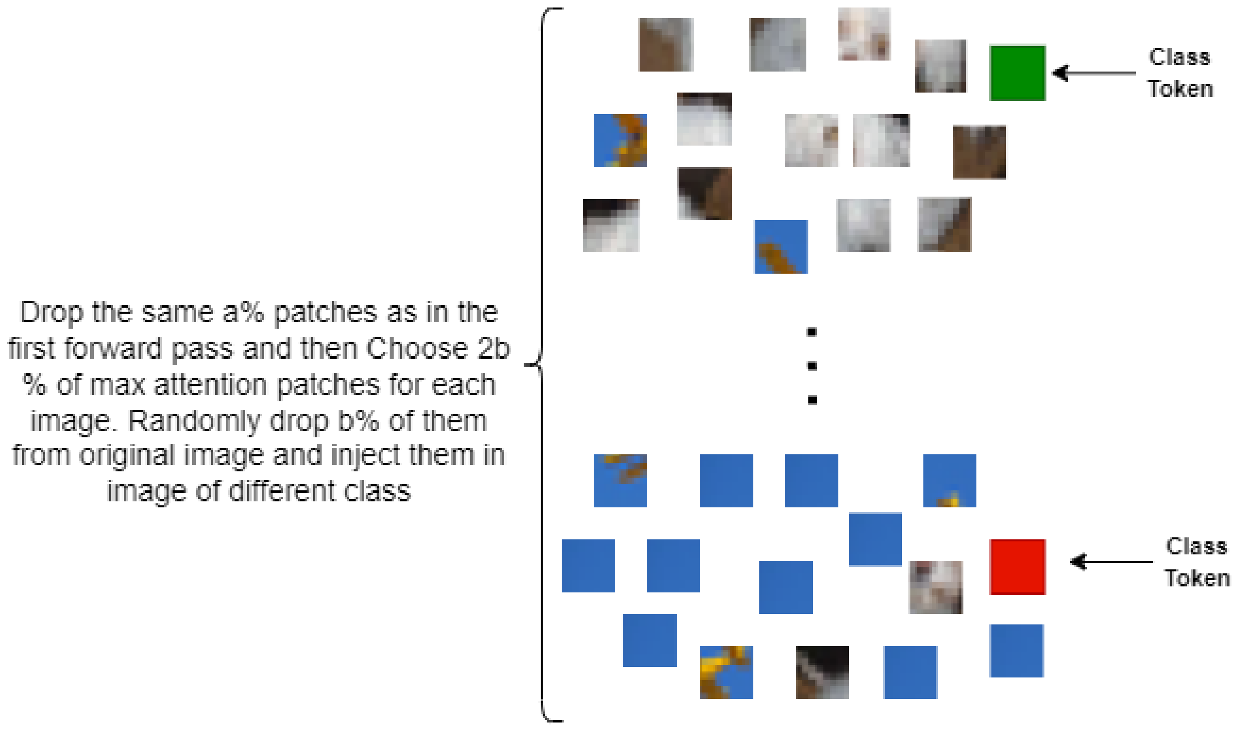

Drop the same a% patches as in the first forward pass and then Choose 2b % of max attention patches for each image. Randomly drop b% of them from original image and inject them in image of different class
Class Token
Class Token

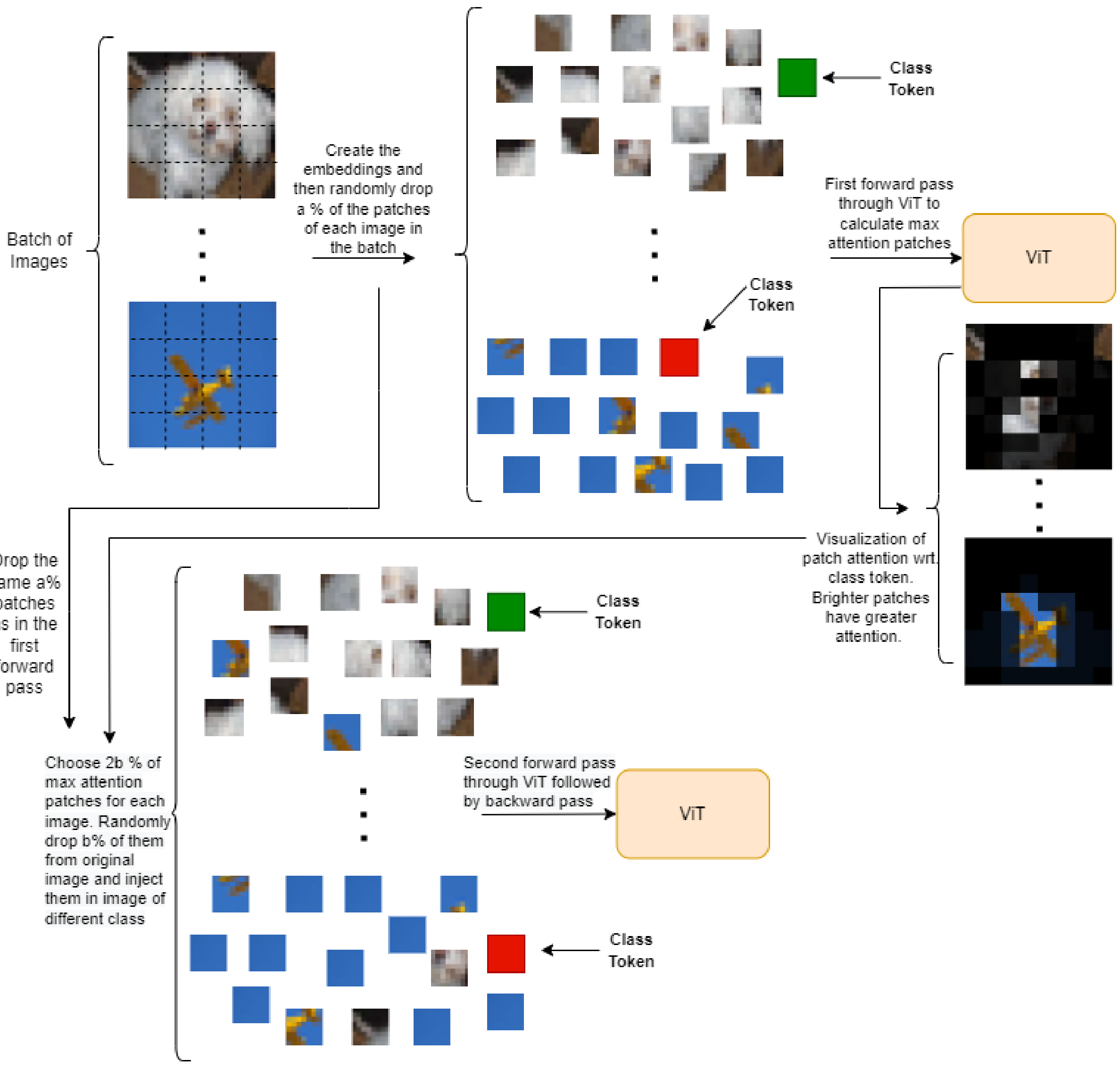
Image 7: Patch mixing visualization. The images shown belong in the CIFAR-10 test batch and are used here purely for demonstration purposes.

Label Mixing:

Label mixing is used in order to properly label the new mixed image samples. Each label of the image pair contributes to the label of the mixed image by a different weight. The weight for the source image label is $\frac{amount\ of\ patches\ of\ source\ image}{total\ amount\ of\ patches\ of\ image}$ and the weight for the target image label is $\frac{amount\ of\ patches\ of\ target\ image}{total\ amount\ of\ patches\ of\ image} = 1 - \frac{amount\ of\ patches\ of\ source\ image}{total\ amount\ of\ patches\ of\ image}$.

Masking b% of the maximum attention patches:

Usual mixing augmentation techniques only have one step of random dropping of patches. In our case, we decide to also drop b% of the maximum attention patches. Experiments show that not only is it computationally efficient but it might also be beneficial for training because some of the maximum attention patches are hidden and the model has to learn from less discriminative patches.



Batch of Images
Create the embeddings and then randomly drop a % of the patches of each image in the batch
Class Token
First forward pass through ViT to calculate max attention patches
ViT
Class Token
Visualization of patch attention wrt. class token. Brighter patches have greater attention.
Drop the same a% patches as in the first forward pass
Class Token
Choose 2b % of max attention patches for each image. Randomly drop b% of them from original image and inject them in image of different class
Second forward pass through ViT followed by backward pass
ViT
Class Token

Figure 3: Entire pipeline of the best mixing augmentation method. The images shown belong in the CIFAR-10 test batch and are used here purely for demonstration purposes.

5. Experiments

All 10 augmentation methods are tested on the CIFAR-10 and CIFAR-100 classification datasets.

5.1 Experiment Settings

The experiment settings regarding the model and training hyperparameters as well as the data augmentation methods tested are recorded below in detail.

Implementing and training the original ViT architecture with hyperparameters:

- *image_size = 3x32x32 (CIFAR)*
- *patch_size = 4x4* → *num_patches = 64*
- *emb_dim = 128*
- *mlp_dim = 4* emb_dim = 4 * 128 = 512 (following the literature)*
- *depth = 6 (6 attn and MLP blocks)*
- *heads = 8*
- *BATCH_SIZE_TRAIN = 128*
- *BATCH_SIZE_TEST = 100*
- *N_EPOCHS = 300*
- *dropout layers are used in the models (dropout probability = 0.1)*

Position embeddings:

- *fixed 2D sin-cos position embeddings (originally used standard learnable 1D position embeddings, but fixed 2D improved model performance - experimentally tested for best setting)*

Optimizer and learning rate scheduler:

- optimizer = torch.optim.Adam(modelc10mix.parameters(), lr=0.001)
- scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, N_EPOCHS)

Training and evaluation data transforms:

```
train_transform = torchvision.transforms.Compose([
   torchvision.transforms.RandomCrop(32, padding=4),
   torchvision.transforms.Resize(sizei),
   torchvision.transforms.RandomHorizontalFlip(),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])

test_transform = torchvision.transforms.Compose([
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize((0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)),
])
```

Label Mixing:

Originally label mixing wasn't implemented at all, but the introduction of label mixing helps in all settings tested. So, only one setting is recorded without label mixing in order to observe how including it increases performance. The labels are mixed accordingly at loss function, depending on how many patches come from the original image (original class) and how many patches come from the image being mixed (mix class). Of course settings 0 and 1 described below require no label mixing.

Recording Results - Training:

- We are training 5 times for each setting on each dataset and averaging the best accuracies recorded in each run to generate the final experiment results.

Augmentation Settings:

General idea of the augmentation method for most of the settings (small differences for some of the settings), 2 forward passes - 1 backward pass:

Drop a % of random patches (random shuffle) from the original image and then forward pass the rest of the patches. Calculate attention matrix, find b% maximum attention patches wrt. class token for each image. For each image, drop b% of patches from the original image (have now dropped a+b % of the original image) and mix the maximum attention patches calculated to another random image of the batch which is of different class. Forward pass, calculate loss, backpropagate.

- a and b are usually both 12.5% of original image ($12.5 \cdot \frac{64}{100} = 8$ patches each)

All data augmentation methods are applied with a probability of 0.5, so the input of the ViT is alternating between batches of only full original images and batches of only augmented images. This specific value has been reached through training experiments.

Setting 0:
Baseline, Original images only.

Setting 1:
Randomly drop 12.5 % patches of original image, then drop 12.5 % of maximum attention patches of original image. No mixing, only patch dropping.

Setting 2: (Without label mixing - new samples keep the label of the original image)
Randomly drop 12.5 % patches of original image, then drop 12.5 % of maximum attention patches of original image. Finally, inject these 12.5 % maximum attention patches to image of different class

Setting 2(lm): (With label mixing - new samples get a label through label mixing)
Randomly drop 12.5 % patches of original image, then drop 12.5 % of maximum attention patches of original image. Finally, inject these 12.5 % maximum attention patches to image of different class

Setting 3:
Randomly drop 12.5 % patches of original image, then drop 12.5 % of minimum attention patches of original image. Finally, inject 12.5 % maximum attention patches to image of different class

Setting 4:
Randomly drop 12.5 % patches of original image, hold random 12.5 % patches to inject to image of different class,then drop 12.5 % maximum attention patches of the original image

Setting 5:
Randomly drop 12.5 % patches of original image, then choose 25 % of maximum attention patches of original image and randomly choose half (12.5 %) out of these to drop. Finally, inject the same 12.5 % maximum attention patches to image of different class

Setting 6:
Randomly drop 12.5 % patches of original image, then choose 25 % of maximum attention patches of original image and randomly choose half (12.5 %) out of these and inject them to image of different class

Setting 7:
Forward pass the original image, then choose 25 % of maximum attention patches of the original image and randomly choose half (12.5 %) out of these to drop. Randomly drop another 12.5 % of the original image, finally inject the selected 12.5 % maximum attention patches to image of different class

Setting 8: (Same as Setting 5 but with 1D Learnable Positional Embeddings instead)
Randomly drop 12.5 % patches of original image, then choose 25 % of maximum attention patches of original image and randomly choose half (12.5 %) out of these to drop. Finally, inject the same 12.5 % maximum attention patches to image of different class

Setting 9:
Randomly drop 12.5 % patches of original image, then choose 25% of maximum attention patches of original image and randomly choose 12.5 % out of these to drop, finally inject 12.5 % random patches to image of different class.

5.2 Results

Below we present the results of all data augmentation methods. In the section "Visualization Results" we also present examples of where the model pays attention for our best setting on the test sets of CIFAR-10 and CIFAR-100.

5.2.1 Quantitative Results

Accuracy table for each setting, For both datasets, Averaging on 5 runs

Settings	CIFAR-10	CIFAR-100
Setting 0	85.89 %	59.63 %
Setting 1	85.46 %	59.17 %
Setting 2	86.70 %	60.61 %
Setting 2(lm)	87.70 %	61.80 %
Setting 3	87.55 %	61.62 %
Setting 4	87.71 %	62.10 %
Setting 5	87.87 %	62.34 %
Setting 6	87.73 %	62.50 %
Setting 7	87.77 %	62.29 %
Setting 8	87.12 %	62.11 %
Setting 9	87.67 %	62.29 %

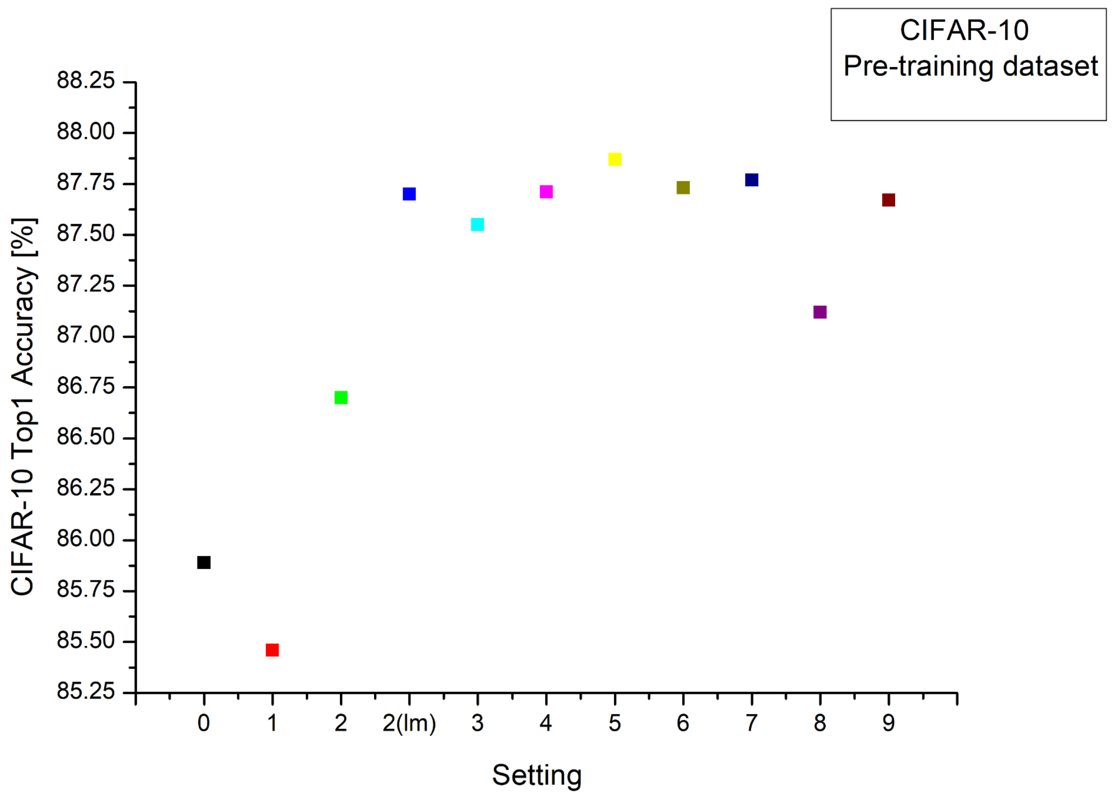


Figure 4: Plotting the settings' accuracies for CIFAR-10 training and evaluation.

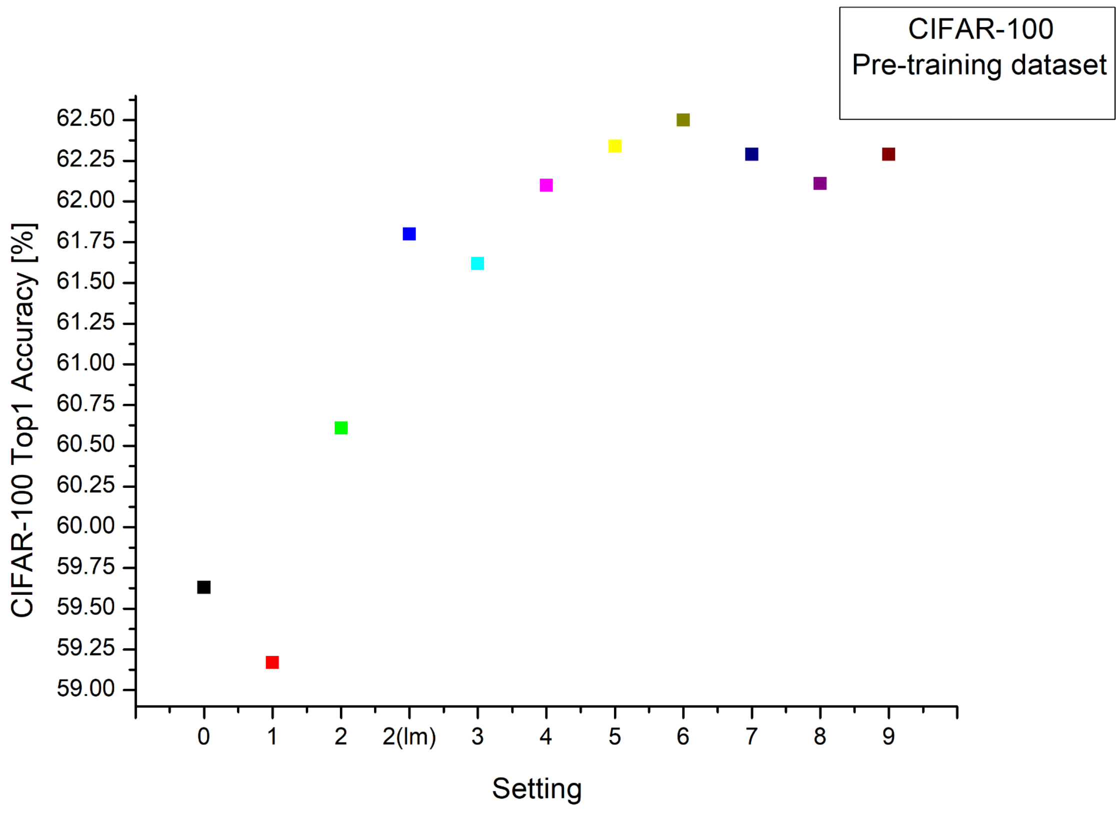


Figure 5: Plotting the settings' accuracies for CIFAR-100 training and evaluation.

5.2.2 Visualization Results

<u>Visualizing attention for best augmentation method model (setting5):</u>

Brightest patches → Patches with greatest attention scores with respect to Class token after last attention layer → Patches most responsible for final image representation and thus final model prediction.

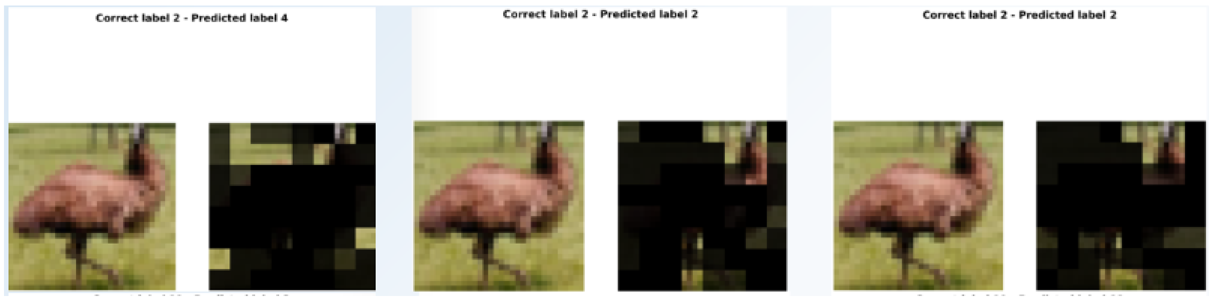

Correct label 2 - Predicted label 4
Correct label 2 - Predicted label 2
Correct label 2 - Predicted label 2

Figure 6: For CIFAR-10 test set image, In epoch 1, 150 and 300 of training

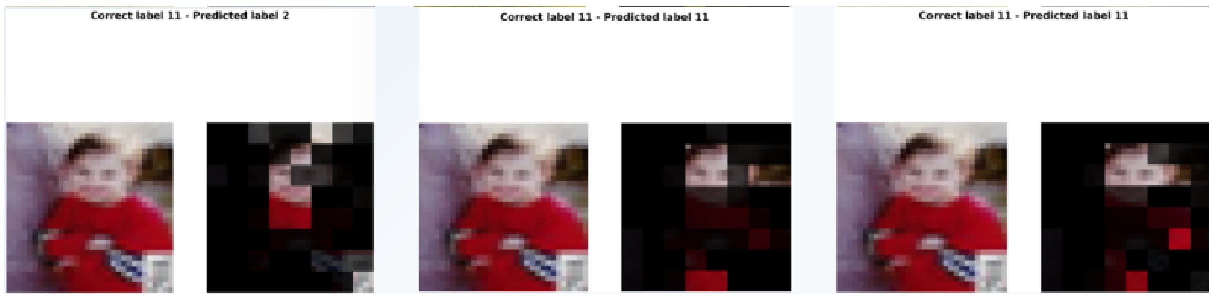

Correct label 11 - Predicted label 2
Correct label 11 - Predicted label 11
Correct label 11 - Predicted label 11

Figure 7: For CIFAR-100 test set image, In epoch 1, 150 and 300 of training

5.3 Results Discussion

Setting 0, the baseline model, outperforms only setting 1 (drop augmentation).
All mixing augmentation methods tested in this thesis improve the baseline model accuracy by a significant margin.

Comparing Settings:

- Setting 2(lm) outperforms Setting 3 → Setting 3 drops b% minimum attention patches and as a results doesn't force the model to attend and learn from less distinct patches of the original image

- Setting 4 outperforms setting 2(lm) → Setting 4 mixes random b% patches so normally we would expect setting 2 to work better. The reason why setting 4 outperforms setting 2 is probably the lack of variety in samples created by setting 2 (lack of randomness). Setting 4 has large deviation in its accuracy (unstable training) between runs as a result of label noise.

- Setting 5 is the best augmentation for CIFAR-10 and second best for CIFAR-100. By choosing 2b% max attention patches and then randomly holding half of it to drop and mix, it introduces more variety of samples compared to setting 2. Additionally, it keeps mixing with high attention patches, unlike setting 4.

- Setting 6 is the only method that outperforms Setting 5 in CIFAR-100. Setting 6 doesn't drop the additional b% max attention patches before the second forward pass and this might be beneficial for datasets like CIFAR-100 with small amounts of data compared to the number of classes. We generally believe that dropping the extra b % patches is beneficial as it reduces computation without deteriorating the model performance. It can even improve accuracy depending on the dataset.

- Setting 7 underperforms compared to setting 5 → Setting 7 doesn't drop a% of patches before the first forward pass and as a result the chosen most salient patches to drop and mix don't have the same variety as in other methods.

- Setting 9 underperforms compared to setting 5 unlike the comparison between setting 2(lm) and setting 4. By introducing some randomness to the maximum attention patch selection we alleviate the problem of not having enough variety of samples.

Ablation studies:

- Setting 8 significantly underperforms compared to setting 5. Setting 8 replaces fixed 2D sin-cos positional embeddings with 1D learnable positional embeddings. It is apparent that fixed 2D sin-cos positional embeddings are better.

- Setting 2 significantly underperforms compared to setting 2(lm). Setting 2 removes label mixing and instead uses the label of the original image for the new data sample. It is evident that proper label mixing is necessary for accuracy improvement.

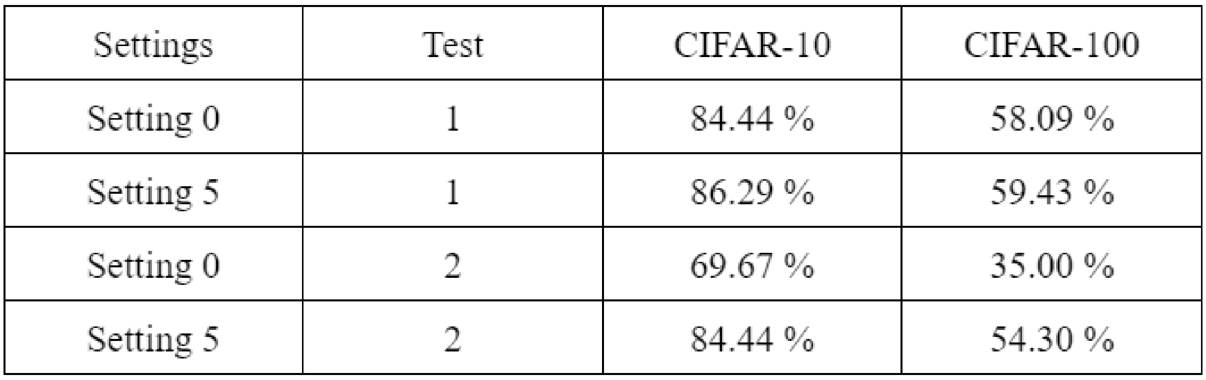
5.4 Noisy Data Test Results

Tested baseline and setting 5 (best augmentation setting) on completely new, unseen noisy data from CIFAR-10 and CIFAR-100.

Test 1: Drop 25 % random patches from each image

Test 2: Drop 25 % random patches from each image and mix these 25 % patches to other image of the batch

Accuracy table for both settings, datasets and tests



Settings	Test	CIFAR-10	CIFAR-100
Setting 0	1	84.44 %	58.09 %
Setting 5	1	86.29 %	59.43 %
Setting 0	2	69.67 %	35.00 %
Setting 5	2	84.44 %	54.30 %

Setting 5 significantly outperforms the baseline model. Setting 5 remains close to its regular accuracy for both types of noise while setting 0 works decently for test 1 but completely fails in test 2 with its accuracy dropping by almost 20%.

6. Conclusions

6.1 Thesis Conclusions

This thesis presented multiple ViT-specific data augmentation techniques, mainly focused on saliency patch mixing, a type of augmentation technique that hasn't yet been used for ViTs. By thorough experimentation, the best augmentation setting yielded very promising results, outperforming the Baseline model on the CIFAR-10 and CIFAR-100 classification benchmark datasets by 1.98 % and 2.71 % respectively.

6.2 Future Directions

Future research should be done in order to better support the benefits of these augmentation methods for the ViT:

- Testing more values to optimize a and b parameters for training with label mixing as they have been originally optimized for training without it

- Comparing our ViT-specific method with other general purpose strong mixing augmentation methods mentioned in this thesis such as Cutmix, SaliencyMix and Mixup

- Testing the augmentation methods on larger models and larger and different datasets such as ImageNet100 and Imagenet-1K

Bibliography

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin. Attention is all you need. In NeurIPS, 2017.

[2] Ilya Sutskever, Oriol Vinyals, Quoc V. Le. Sequence to Sequence Learning with Neural Networks. In Advances in Neural Information Processing Systems, pages 3104–3112, 2014.

[3] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N Dauphin. Convolutional sequence to sequence learning. In ICML, 2017.

[4] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. arXiv preprint
arXiv:1607.06450, 2016.

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 770–778, 2016.

[6] An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale
Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil Houlsby, In ICLR 2021 Oral

[7] Better plain ViT baselines for ImageNet-1k. Lucas Beyer, Xiaohua Zhai, Alexander Kolesnikov arXiv:2205.01580

[8] Xinlei Chen, Saining Xie, and Kaiming He. An empirical
study of training self-supervised vision transformers. In International Conference on Computer Vision (ICCV), 2021

[9] Maithra Raghu, Thomas Unterthiner, Simon Kornblith,
Chiyuan Zhang, and Alexey Dosovitskiy. Do vision transformers see like convolutional neural networks? CoRR, abs/2108.08810, 2021.

[10] Randall Balestriero, Ishan Misra, Yann LeCun. A data-augmentation is worth a thousand samples: Exact quantification from analytical augmented sample moments.
arXiv:2202.08325

[11] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1:541–551, 1989.

[12] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, Lei Zhang; Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2021, pp. 22-31

[13] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, Hervé Jégou. Training data-efficient image transformers & distillation through attention. In PMLR, 2021

[14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. International journal of Computer Vision, 2015.

[15] K He, X Chen, S Xie, Y Li, P Dollár, R Girshick. Masked autoencoders are scalable vision learners. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recongition, 2022

[16] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In AAAI, 2020.

[17] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. arXiv preprint arXiv:1909.13719, 2019.

[18] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: ´ Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412, 2017.

[19] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. arXiv preprint arXiv:1905.04899, 2019.

[20] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. arXiv preprint arXiv:1708.04552, 2017.

[21] Uddin, A., Monira, M., Shin, W., Chung, T., Bae, S.H., et al. Saliencymix: A saliency guided data augmentation strategy for better regularization. arXiv preprint arXiv:2006.01791 (2020)

[22] Walawalkar, D., Shen, Z., Liu, Z., Savvides, M.: Attentive cutmix. An enhanced data augmentation approach for deep learning based image classification. arXiv preprint arXiv:2003.13048 (2020)

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. University of Toronto, 05 2012

[24] Kumar Singh, K., and Jae Lee, Y.. Hide-and-seek. Forcing a network to be meticulous for weakly-supervised object and action localization. In ICCV, 2017

[25] Jie-Neng Chen, Shuyang Sun, Ju He, Philip HS Torr, Alan Yuille, Song Bai. Transmix: Attend to mix for vision transformers. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2022

[26] Jihao Liu, Boxiao Liu, Hang Zhou, Hongsheng Li, Yu Liu. TokenMix: Rethinking Image Mixing for Data Augmentation in Vision Transformers. arXiv:2207.08409 (2022)

[27]https://medium.com/intro-to-artificial-intelligence/semantic-segmentation-udaitys-self-driving-car-engineer-nanodegree-c01eb6eaf9d

[28] Md. Rakib Hossain Khan, in https://www.semanticscholar.org/paper/Deep-learning-based-medical-X-ray-image-recognition-Khan/fdd644ec07a4e54d1860825f07f74da2f6f8e442

[29] https://azati.ai/image-detection-recognition-and-classification-with-machine-learning/

[30] Abhishek Singh, in https://www.kaggle.com/ganu1899