Explore and Implement Image Augmentation Techniques

[1]Matcha Pavan Kumar, [2]Priyangshu Sarkar, [3]Arpit Mahajan, [4]U. Hariharan
1,2,3,4: - Department of Computer Science and Engineering, APEX Institute of Technology, Chandigarh University, Punjab, India.
Email: 20bcs6072@cuchd.in, 20bcs6047@cuchd.in, 21bcg1115@cuchd.in, hariharan.e11201@cumail.in

Abstract— ***For a variety of computer vision tasks, Convolutional Neural Networks (CNN) have exhibited astounding performance. From self-driving cars to disease detection and prediction, CNNs proved to be state-of-the-art in the field of image processing. However, the drawback of any deep learning network is the data. The performance of the model depends on the quantity, quality, and distribution of the training dataset. Training on smaller datasets can result in overfitting. Sadly, many domains are limited in terms of data like the medical industry, etc. The solution to this problem is Data Augmentation. It is a data-space solution to the issue of limited data. The term "data augmentation" refers to a group of methods used to increase the amount and caliber of training datasets so that better deep-learning models can be created with their aid. In this paper, we propose six new augmentation techniques that are designed to guide the model to improve its edge detection, texture analysis, better feature representation, and improving its inference capabilities. The proposed techniques are efficient as they are adaptive to the current model's understanding of the data and improve the areas where it is not up to the mark. These six augmentations are tested with custom test designs which are developed according to the techniques. We have explored the extent to which image augmentations can be used to improve the robustness of the model and in our research, we have concluded that image augmentations may not be the best or practical approach in making a model robust to variations; the changes need to be made at the architecture level of CNN models. So, we are proposing a new idea called "Transferable Model" where the weights learned from one part of the model will be transferred to another. They have inter-model communication capabilities. We have shown how we can use this idea to make a model robust to variations with less training.***

***Index Terms*— Image Augmentations, Feature Maps, Activations, Information Encoding, Adaptive Augmentation Techniques, Transferable Model**

I. INTRODUCTION

Today Convolutional Neural Networks (CNN) have become a dominating technology in the industry related to image processing. Previously, we needed to manually handcraft the features to be detected. That is a tiresome task and a programmer cannot provide all possible variations of a feature[1]. The CNNs exactly address this issue. The architecture of the CNNs has the advantage of automatically learning the important features to be detected based on the guidance of the training dataset. Using backpropagation, a CNN model will improve its feature detection and inference capabilities based on the feedback from the output[2]. If we can provide enough data to CNN models, research has shown that they can achieve human-level performance[3-5]. However, this is the biggest disadvantage of dealing with deep learning models. They are data-hungry technology. They work well only if they have huge data to learn from. If they are trained on smaller datasets, the model will get overfitted to the data. That means, rather than generalizing the features from the data, the network memorizes them. Due to this, the model does not perform well on the test data or unseen data[5]. The only solution to this problem is to provide a larger dataset. Making a dataset is a huge task; we need to collect the data from the real world, manually label them, and also, we need to ensure that we have collected diverse data [6]. Also, in the sectors like medical industry, data is scarce. Due to the high cost of labor and equipment and various patient privacy laws, it is hard to get large amounts of data in those domains. The final solution to all these problems is to artificially inflate the dataset. This is called Data Augmentation.

It is a technique using which we will introduce changes to the original data and treat the modified data as new training data. Using augmentation techniques, we can fix the issues of class misbalancing, introduce diverse representation to the data, ensure that the model is not overfitting [5], and guide the model in the direction we require. We know that the only communication we have with the model is through the distribution of the training dataset. For instance, if there are more black cats in the training data, the model will learn if the entity is black then most probably it is a cat. So, using the data augmentation, we can guide the model as required. We just need to exaggerate the feature we want the model to learn in the training dataset using image augmentations.

We are introducing six new augmentation techniques that can improve the model's understanding of the data. They are named: 1-Pixel Shift + Flip, Activation Suppression,

Adaptive Color Scheme, Tiles Shuffling, Low-Level Residual, and Texture Information Encoding. Each technique is designed to address different issues in a CNN model efficiently. These techniques are designed to have a greater impact on the model's understanding of data with less amount of data generation. We have explained different issues with certain augmentation techniques as well as CNN architectures and how we can address them using these techniques, we have attached the results, analyzed the behavior of each technique, and identified the pros and cons.

The two main goals that we are trying to achieve using Image Augmentations are:
1. To guide the model in the right direction to learn the right features. We can exaggerate the features using augmentations so that the model will have no difficulty detecting those features and guided upon them.
2. To ensure that the model is invariant to variations like translation, scaling, zooming, rotations, colors, etc. Here, invariance is a property of the network to behave similar way on data even if it is subjected to transformations.

In our research, Image Augmentation is the best approach to achieve the first goal (guiding the model in the right direction) but it is not the best approach to achieve the second goal. For example, let's assume that we need to make a model that is invariant to rotations, through Image Augmentation, we would either provide all 360° rotated versions of an image or at least some angles of the image. For a dataset of size 3000, if we rotated every image for all 0° to 360° we will end up with a dataset of size 10,80,000 images. It takes a large amount of time to generate all the data and train the model. The main drawback of this technique is that even after training on such a huge amount of data, it does not guarantee that the model will be 100% invariant to rotations. As a deep neural network is a black-box we cannot estimate if the resultant model is invariant to rotations. If we extend this to translation (providing all possible translation versions of the same image), color (providing all possible shades of the same image), etc. We can understand that Image Augmentation is an impractical approach to making the model robust to all kinds of variations (the second goal).

To achieve the second goal, the most important thing to remember is that the problem is the architecture. Right now, we have an architecture where features are learned automatically based on the distribution of data [7]. The feature representations are very rigid and follow the distribution of the training data. So, if we can have an architecture where the feature representations can be transformed as needed, it solves the problem. We call these models "Transferable Model" where the knowledge from one part of the model is shared with another part of the model. In this type of model, we just need to train the model to gather the required features and after that, we will transform the knowledge as required and transfer it to the required parts of the model. Due to this, we need not train the model explicitly. The idea is that, if the knowledge is transferable, we will just transfer it instead of training the model explicitly. This saves both time and computation. You can find out more about this in the upcoming sections.

II. LITERATURE SURVEY

Image Augmentation is a very popular technique that we have used for many years but it gained popularity with the implementation in AlexNet [8]. They used augmentation techniques on the ImageNet dataset to artificially inflate the data and improve the robustness of the model. After that, we have many categories of Image Augmentation techniques like Geometrical Transformations, Color Space Augmentations, Kernel-based augmentations, Deep Learning based Augmentations, etc [9].

A. 1-Pixel Shift + Flip

As demonstrated by [10], one of the main disadvantages of CNNs is the pooling layers. Due to the pooling layers, we have the benefit of computational efficiency and high-level feature representation but this comes at the cost of loss of information. In pooling operations, it only considers the information of oddly spaced pixels and completely ignores the information of evenly spaced pixels. Due to this, we are losing lots of texture information, which results in an unstable model. The model's predictions fluctuated with even 1-pixel shifts. They addressed this by introducing an architectural change called "blur_pool" where we have layers that blur the feature maps before passing them to any of the pooling layers. Due to this, the information spreads across the image, resulting in less loss of information. But in our approach, we solved this issue by clubbing two augmentation techniques Flipping and Shifting. We performed Flipping as it is one of the basic augmentations to be performed and by performing a 1-pixel shift we placed all the evenly placed pixels in place of the oddly placed pixels.

B. Activation Suppression

The major inspiration for this augmentation technique is an augmentation called Random Erasing [11]. Here, the idea is that in an image we will generate a random coordinate of a box and fill that box with black or grey or random Gaussian noise, etc. The main goal of this technique is to create an occlusion effect on the subject and improve the model's prediction with a limited amount of data. It is indeed a very good technique but due to its stochastic nature, it may not be efficient as sometimes it may cover the subject completely and sometimes it may not even cover the subject (not performing as intended). We have other types of augmentations that draw inspiration from random erasing like Cutout[12], CutMix[13], AugMix[14], Attentive Cutmix[15], etc. Where the erased region is filled with another image. The second image can be of the same class or a different class, the final label of the new image will be decided based on the proportion of the image from each class. Due to this, the convolution operations are not wasted. But this augmentation introduces sharp edges into the image and we know that CNN models are sensitive to edges. So, sometimes these edges take part in the inference of the model.

Also, the representation of the data after performing these augmentations may not represent real-life data. In our approach, we designed the augmentation to adapt according to the model's current understanding of the

data. We do this by extracting the feature maps of the model and suppressing the high-activation regions. We have also taken care not to generate any sharp edges through this augmentation.

C. Adaptive Color Scheme

We perform many pre-processing steps on an image. We will normalize the range, resize them, perform batch normalization[9], etc. In the same way, for tasks that are not color-specific, we need to ensure that the dataset is not imbalanced with colors. For that, we have a technique called Histogram Equalization[16], where we analyze the histogram distribution of each channel and try to find the distribution in such a way that the graph will be closer to a flat line. Due to this, the model will not be sensitive to any color. From a theoretical standpoint, everything is right but when implemented, the Histogram Equalization will not have a flat line graph, rather close to it. Even after applying this, there is no guarantee that the model will be less sensitive to colors. In our approach, we will perform the histogram analysis on each image, invert it, and find the top n shades. Next, we will apply these shades to the image.

D. Tiles Shuffling

One of the disadvantages of CNN as demonstrated by [17] is that they have no sense of the relative spatial placement of the features. They just try to detect the presence of the features. They have no information regarding the relative placement of each feature. [17] have addressed this issue using a new type of network called the Capsule Network. This type of network represents the features using vectors. Using that, they create a sense of entity in the network. But they have their limits; we need to maintain additional information and these networks are computationally expensive due to added parameters. In our approach, we will divide an image into nine parts and shuffle them randomly. And we will introduce this new set of images as a new class. By doing so, we will be converting binary problems into categorical problems. Now, the network will know the distinction between the right placement of features and the wrong placement of features.

E. Low-Level Residual

The inspiration for this technique is taken from ResNet architecture[18] as well as Noise Injection Augmentation [19]. The idea of residual is important here. The main goal of this technique is to improve the edge detection of the model by applying direct kernel-level regularization. Like in ResNet, which learns the residual function, we have a pre-trained model and we pass an image into the model to detect the edges. After that, we will erase the detected pixels and at last, we will end up with a residual. By erasing, we mean to introduce artifacts into the image. This creates a similar effect to noise injection augmentation.

F. Texture Information Encoder

This augmentation also addresses the issue mentioned in the 1-pixel Shift + Flip augmentation. The goal of this technique is to incorporate the information of the evenly spaced pixels onto the oddly spaced pixels. We do this by exchanging the information from evenly to oddly spaced pixels using a weighted sum. With this, we can preserve the information within the original dataset.

Transferable Model: This is a type of CNN architecture where the information from one part of the model is transferred to another part of the model. The inspiration for this architecture is taken from various existing methods like TI Pooling [20], Harmonic Networks[21], Rotational Invariant Coordinate Systems [22], and Spatial Transformer Networks [23]. In TI Pooling, the input image is rotated at the specified degree and finds the degree at which the kernels are activated highest; this causes additional overhead as an image needs to be rotated and predicted. In Spatial Transformer Networks, the goal is to find the canonical form before passing the image into the model for prediction. The top additional layer causes overhead as it needs to find the canonical form of the object. Rotational Invariant Coordinate System is indeed a sound approach but still, we need to find the value of the angle at which there is maximum activation. In our approach, we focus on the feature representation in the networks and after that, we will transfer feature representations with appropriate transformations. Now, we only need to worry about the features rather than the whole image; with the parallel Siamese networks [20], a feature will be detected irrespective of the orientation of other features.

III. Methodology

In this section, we will explain the workings of all six Augmentation techniques as well as the Transferable model. We are using the Cat vs. Dog dataset; the main reason for selecting this particular dataset is that this data is complex with many types of cats or dogs at different lighting conditions, textures, orientations, etc. So, if we can show improvement in this kind of dataset, we can expect similar results in others. The training, validation, and testing splits of a total of 25000 images are shown in Table I. We have limited the training data to a total of 3000 to simulate the condition of data scarcity so that we can analyze augmentation techniques to improve the model's understanding of the data.

TABLE I

Dataset	Training Size	Test Size	Validation Size
Cat	1500	10740	200
Dog	1500	10740	200
Total Data	3000	21480	400

The total size of the original dataset is 25000 out of which there are some corrupted files. We recovered the majority of it and ended up with a total of 24,880 images.

A. 1-Pixel Shift + Flip

The main motive for developing this technique is to address the issue of information loss due to pooling layers. It is demonstrated by [10]. Pooling layers are very important in terms of the efficiency of the model as they can reduce the dimension of the data to deal with

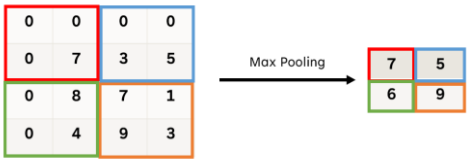
and also help in getting high-level features. In other words, CNNs without pooling layers are very inefficient. But in the pooling operation, due to its non-overlapping sliding operation, the information in the evenly spaced pixels is completely neglected as demonstrated in Fig. 1. In tasks where texture plays a major role, due to the nature of pooling layers, the model will be unstable even if the image is shifted by a small number of pixels. We demonstrated the instability in the data by performing max pooling operations for 3-levels on a normal image and a 1-pixel shifted image (refer Fig. 2 and Fig. 3) and we derived a deviation graph (refer Fig. 4) showing the number of pixels deviated and to what extent. We solve this problem by shifting all the images by 1 pixel both in horizontal and vertical directions. Due to this, all the evenly spaced pixels are now placed in odd places. Now, the model with the current pooling approach will have all the information it needs to work. But, generating new data with just a 1-pixel shift may not be efficient in terms of resources and time. So, we have clubbed this with Flip Augmentation. As flipping is one of the basic augmentations we need to perform, we have clubbed the 1-pixel shift with flipping. But, we can also club this with other techniques. This technique is efficient as we are ensuring the model gets all types of information as well as the kernels are made symmetric.

Steps:

1. Shift the image by 1 pixel both in the horizontal and vertical directions.

2. Flip the image around the x-axis.

(a)



(b)

Fig 1: a) Pooling without 1-pixel shift (computing on oddly placed pixels), b) Pooling with 1-pixel shift (computing on evenly placed pixels). By shifting 1 pixel the output of the pooling operation is changed.

Fig 2: - After applying the max pooling operation on a 300x300 image.

Fig 3: - Applying the max pooling operation on the same image but with a 1-pixel shift in both horizon and vertical directions.

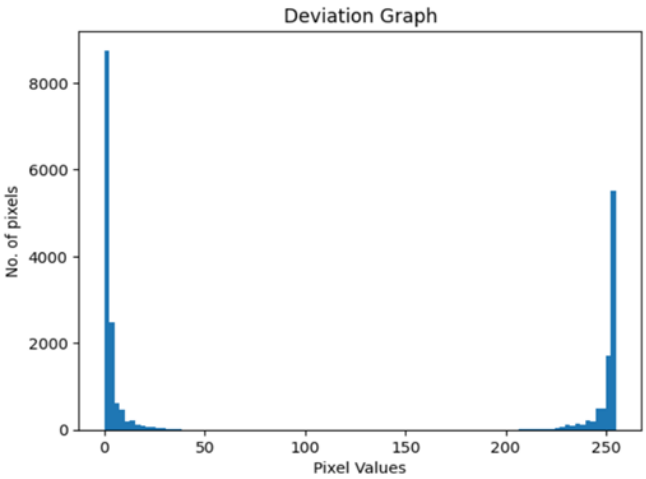


Fig 4: - This plot shows the number of pixels deviated and by what amount between the outputs of pooling operations on the original image and the 1-pixel shifted image.

We can see that out of 22500 (150 X 150) pixels, about 8500 pixels have zero deviation, and all the rest of the pixels are deviated to some extent.
Total: (22500-8500)/(22500) = 62.22% of the pixels deviated to some extent with just a 1-pixel shift.

Advantages:

- By shifting the images by 1 pixel, we are easily ensuring that the model will get all pixel values to learn. And with the inclusion of flipping, we are efficiently getting the benefits of both.
- As these are just based on image modifications, the resource requirement is very low.
- It is a very simple augmentation technique. We just have to perform shift and flip operations. So, the complexity of coding is low.
- It improves the robustness of the model to the pixel-level variations.

Disadvantages:

- If there is not much change in the pixel information of even and odd places, we may not see any impact on the performance.

- If texture detection is not important, then this augmentation is not the appropriate one.
- Using this technique without clubbing with others can be inefficient.

B. Activation Suppression

This technique is used to suppress the activated regions in the final layer of the feature maps [24] so that the model is forced to look at some other regions on the image. We can say that this is a kind of feature regularization technique. In this technique, we will maintain a base model that is trained on the original data. Next, we will pass each image from the training data (refer Fig. 5a) to the model, extract the feature map (refer Fig. 5b), and find the top n activation regions in the image and suppress them (refer Fig. 6b).

Here, by suppression, we mean that we will fill those regions with neutral colors like gray (It is the same as a paintbrush in Photoshop with diffused edges) and also, we will apply a stylizing effect like oil painting, cartoon, etc so that the model is forced to understand the structure rather than pixel-level information. This results in improved detection of the subject in the frame (refer to Fig. 7b).

The main advantage of this technique is that it is adaptive based on the current model's understanding of the subjects in the image. We are blocking the features that the model already knows so that it can learn other features.

So, this technique is very efficient as whatever new data is generated is based on the performance of the model, and we are only teaching the model where it is lagging. Another pattern we found is that it always has a positive effect on the model [25]. If we suppress the background regions, the model is directed to look for a suitable subject, and if we suppress some parts of the subject, then the model is directed to look for some other features of the subject [26]. Also, we are improving the feature representation in the network, which reduces the black box effect of a CNN network.

This augmentation technique is designed to impact a CNN model in two areas:

1. Improved Detection (also results in improved feature representation in the networks).

2. Improved Inference Capabilities (As the model is forced to infer based on limited details).

This can be used for active feature regularization as this augmentation directly impacts the feature detection and inference of the model.

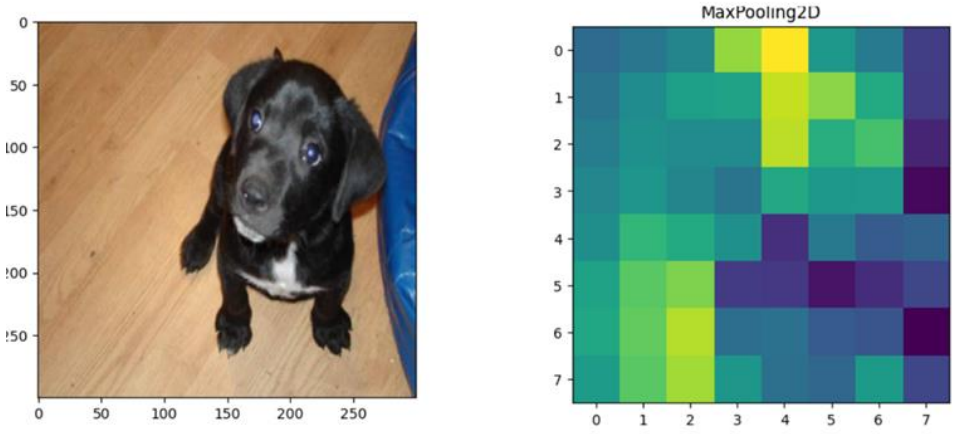
Steps:

1. Develop a pre-trained model on the original dataset.

2. Pass each image from the training dataset and extract the feature map of the last convolution or pooling layer.

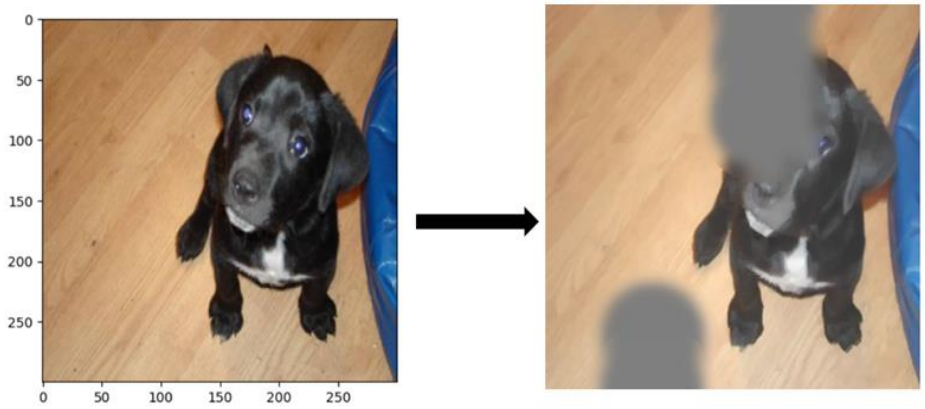
3. Select the top n (2 – 5) regions in the map and suppress them using a neural color in a circle with diffused or blurred edges.

4. Apply any stylizing effect like oil painting, cartoon, etc.



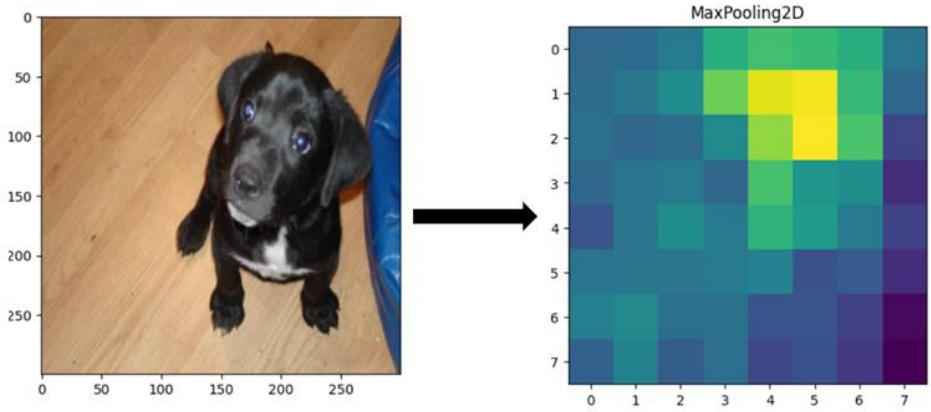
(a) (b)

Fig 5(a, b): - (a) Original image (b) Feature map of the original image. We can see that the model is also activating in regions where the subject is not present.



(a) (b)

Fig 6(a, b): - (a) Original image (b) Suppressing the activations by filling the areas of activations with neutral colors like gray. And to make the model treat this image as new we have applied an oil paint stylizing effect.



(a) (b)

Fig 7(a, b): - (a) Original Image (b) Improved feature detection in terms of detection and accuracy.

Advantages:

- Dynamic Augmentation technique. This means, the changes are made to the original data based on the current model's understanding.
- No wastage of resources as we are giving information to the model that it did not know.
- Improves the feature representation of the image. This reduces the black box effect of the neural network.
- Provides automatic attention to important features.
- It also creates the effect of occlusion. Due to that, it improves the inference mechanism of the model.

Disadvantages:

- If the model has already learned the features, then this technique produces no effect.
- If the subject is very small i.e. less than the size of the suppression circle, it produces images with no subject. This can reduce the model's performance.
- We need a base model to apply this technique.
- If the model is deeper, the time and resource consumption for data generation will be high.

C. Adaptive Color Scheme

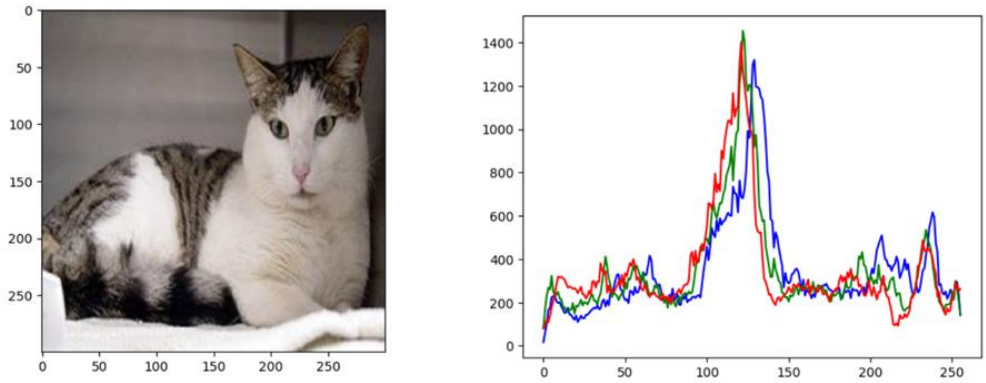
The main motive for using this technique is to make the model robust to color variations. The color distribution of the training dataset will have an impact on what colors the model is going to look for. For example, if there are more orange cats in the dataset, then the model might believe that cats are only orange. So, to address these kinds of issues we have the adaptive color scheme. Here, we will analyze the color distribution of each image in the training dataset using a histogram (refer Fig. 8a,b) and inverse it so that the colors that are not in the images pop up (refer Fig. 9a,b). Now, among these colors, we will select the top 6 colors and apply that color shade to the image to produce a new image with different shades (refer to Fig. 10a-f).

The main advantage of this technique is that we are not producing any shade randomly rather we are producing the worst-case scenario of shades. Due to this, the thresholds in the network are tuned to be activated for a wide range of shades. In other words, these particular shades will have more impact on making the model robust to colors than any other shades.

Steps:

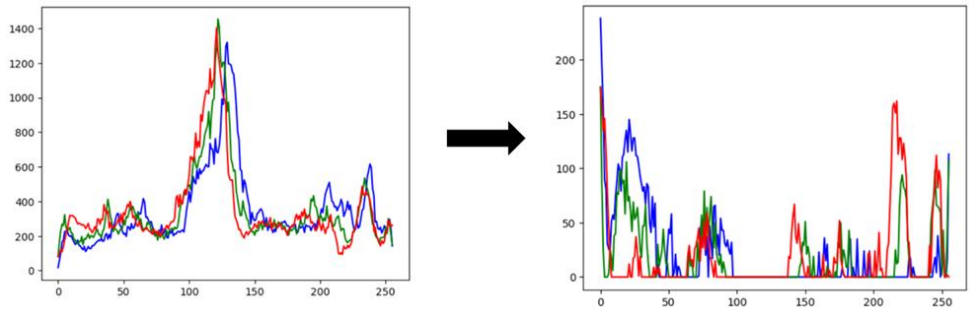
1. Get the histogram distribution of each image.
2. Invert the distribution and let the minimal colors come to the top.
3. Find the top n shades among them.
4. Apply that particular shade to the original image.

This is an efficient technique to improve the model's invariance to colors. Rather than providing every shade, we are providing the shades that are present in the least amount. Due to this, the kernels at the top layers have new information about the relative R-G-B order. This ensures maximum impact on the model.



(a) (b)

Fig 8(a, b): - Histogram distribution of each channel in an image



(a) (b)

Fig 9(a, b): - Inverting the original histogram distribution

(a) (b) (c)

(d) (e) (f)

Fig 10(a-f): - Using the inverted histogram, the top 6 shades are derived that are present minimal in the image. In this way, we provide the worst-case scenario to the model.

Advantages:

- Dynamic Augmentation technique. So, the new shades that are produced are according to the input image.
- The generated shades will have a high impact on the model as the model did not see this shade. So, it is an efficient augmentation technique.
- Rather than providing many shades to make the model color invariant, we will provide top shades that can impact the model towards making it color-invariant

Disadvantages:

- It will make the model color invariant to some extent but it is not truly color invariant.
- There is no direct method to decide on the number of shades we will produce. Sometimes, we may produce shades that the model already knows. In such cases, it will not have any impact.

D. Tiles Shuffling

In this technique, we will divide an image into nine parts and shuffle these randomly (refer to Fig.12a-d). Per each image, there are a total of 9! combinations we can generate. To ensure that the shuffling does not produce any sharp edges, we will smooth or blur those regions.

This technique is developed to address two different issues in a CNN model:

1. The problem of Binary Classification:

In Binary Classification, we have one output neuron. The output of 0 represents class A and the output of 1 represents class B. Let's assume a scenario where we have given an image of an object that belongs to neither of the classes. In that situation, even though the object does not belong to any class the model most probably categorizes it into any one of the classes, and it is not guaranteed to get 0.5 as output every time. Due to this, it is harder to test the model's performance for each class.

This is because, if the model does not find the feature of class A, then it will directly classify the object as class B irrespective of whether the required features are present for class B or not. The solution to this is to convert the problem into a categorical one by introducing a new class. So, using the Tile Shuffle technique we will generate images of that extra class.

2. The problem of the CNNs to understand the relative placement of features:

As demonstrated [17], we know the CNNs do not have any information about the relative placement of features. CNNs only try to check if the required features are present or not; they don't maintain any information on the placement or orientation of the features. So, if you provide a disfigured image of a human face, it will predict that the image is human (refer to Fig. 11). So, the model does not have a sense of entity. So, using this technique we can address this issue to some extent. By shuffling the features of the subject and treating them as a new class, we are ensuring that the model knows the distinction between the right placement of the features and the wrong placement of the features. Even if the subject is oriented at different angles this augmentation technique tries to capture the relative placement of features.

Steps:

1. Divide the image into 9 squares.
2. Randomly select two numbers between 1 – 9 and swap those tiles.
3. To avoid sharp edges, blur the regions of sharp changes.

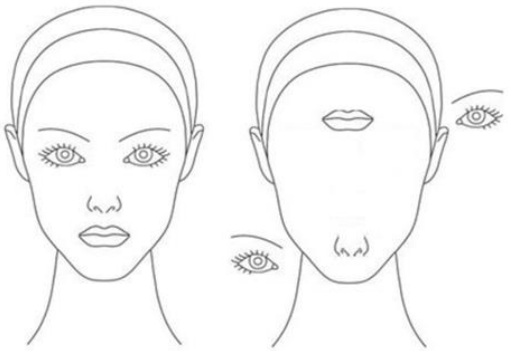
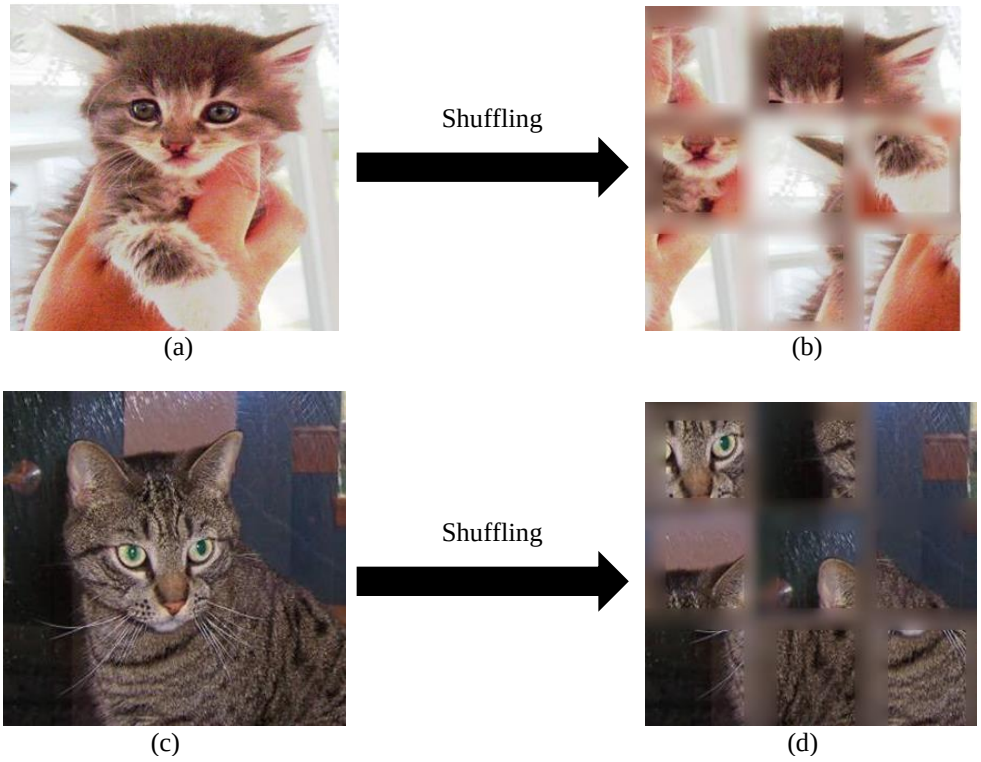


Fig 11: - A CNN model that is trained to detect human faces will predict "human" in both cases.



(a) (b)

(c) (d)

Fig 12(a - d): - Using Tile Shuffling, we will reorder the relative placement of the features so that the model knows the distinction between the right order and the wrong order.

Advantages:

- Using this technique, we can convert a Binary classification problem into a Categorical classification problem. As we know the issues of the Binary classification problem.
- It improves the model's understanding of the relative spatial information of different features.
- There are a total of 9! combinations possible for each image. So, virtually, we can generate as much data as we want using this technique. This is an advantage because we need the data to be balanced among the classes.

Disadvantages:

- It is harder to test the impact on the model's performance with this technique.
- If there is no priority for spatial information for a given task, then this technique is not the appropriate one.

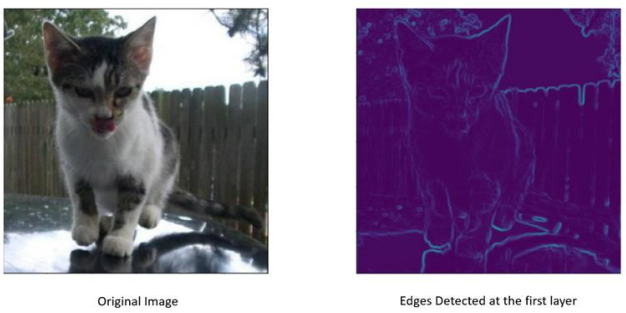
E. Low-Level Residual

This technique is inspired by ResNet [18]. As the name suggests this technique is designed to improve edge detection at the top layers. In this technique, we will extract the feature map of the first convolution layer (refer to Fig. 13a,b), cover the regions where edges are detected with a neutral color, and show only the non-detected part. Due to this, the model will look for new types of edges. We will cover or hide the regions where the convolutions of a kernel are producing high values (refer to Fig. 14a, b). We will exactly fill the set of pixels where the convolution is done [27]. So, we will be modifying it at the pixel level. Due to this, the kernels are directly regularized and forced to learn new types of edges (refer to Fig. 15a, b). We have demonstrated the deviation in the feature maps before and after applying this augmentation in Fig. 16a-c.

In this technique, after generating the low-level residuals, we will freeze all the layers except the top layers. Due to this, the changes in the top layers can be seen directly at the output. This mitigates the vanishing gradient problem. And also, as we have fewer training parameters, the resource and time consumption are lower. As we are introducing artifacts into the image, from one perceptive, we are introducing noise into the image. So, this technique will also provide the impact of noise injection augmentation [19].

Steps:

1. Develop a pre-trained model on the original dataset.
2. Get the feature map of the edge detected from the first layer of the network.
3. Identify the pixels that are highly activated and fill all the pixels that participated in that particular convolution with a neutral color like gray.



(a) (b)

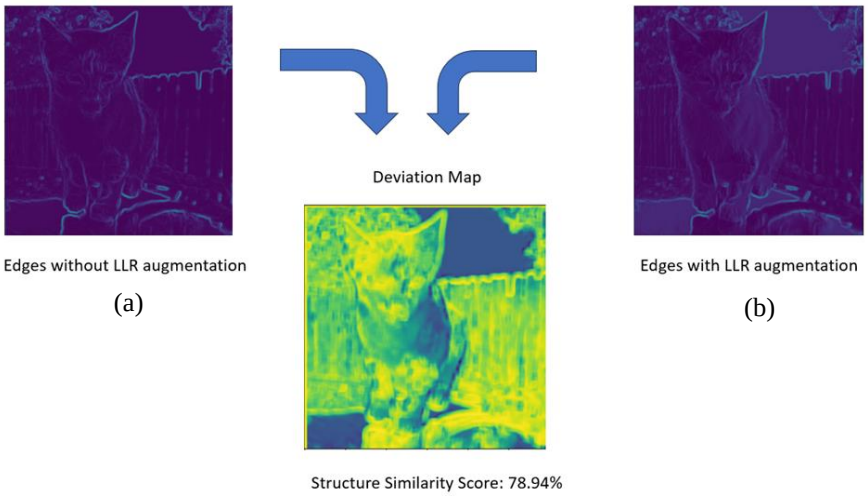
Fig 13(a, b): - This is the original image and edges detected using the first layer

(a) (b)

Fig 14(a, b): The left image is the original image and the right image is the new image after applying the Low-level Residual augmentation.

(a) (b)

Fig 15(a, b): - This is the new feature map of the model trained on the new images



(a) (b) (c)

Fig 16(a-c): - (a) Edges without Low-Level Residual augmentation (b) Edges with Low-Level Residual augmentation (c) Structure Similarity Score between the two maps: 78.94% which is shown in the Deviation map. This shows a 21.04% improvement in the detected regions.

Advantages:

- Dynamic Augmentation technique. Here, we are injecting the artifacts based on the model's current understanding of edges.
- As the artifacts are at the pixel level, the kernels are directly regularized.
- As only the top layers are trained, we have fewer training parameters. So, we have efficient usage of resources, and also, the changes in the top layers have a direct impact on the output. Due to this, we don't face any vanishing gradient problems.
- We currently, implemented this for low-level features but we can also extend it to high-level features.

- As we are producing artifacts at the pixel level, from one perspective we are injecting noise into the image. So, this augmentation can improve the model's robustness to noise.

Disadvantages:

- If the model already knows the detection of all types of edges, this augmentation will not have any impact.
- Overtraining with this data can harm performance.
- We need to have a base model to apply this technique.

F. Texture Information Encoding

This technique is developed as an alternative to the 1-pixel shift + flip augmentation technique. In this approach, we will address the issue directly at the root level. In this technique, we will try to incorporate the information from the evenly spaced pixels onto the oddly spaced pixels as represented in Fig. 17. By doing this, we can ensure that the information is still preserved even after performing max pooling. The impact of this technique can only be seen if the image has some texture information. In other cases, it may not have much impact.

Steps:

1. Lower the brightness of the image. When adding the neighboring pixels, the value of individual pixels automatically increases and this may result in an extremely bright image. To avoid that, we will reduce the brightness at the beginning.

2. The values of the even places are added to the values of the odd places as specified in Fig. 17. We have a parameter to decide on the proportion of information to be added.

3. We do this for all rows and columns. In this way, we can incorporate information from the even pixels into the odd pixels.

Fig. 19 is the encoded image. It will be similar to the original image (refer to Fig.18) as the goal is to maintain the same information but include the maximum information in the pooling regions. Due to this, without generating new data, we can encode the information in the same image and the model trained on this data will have more information to learn.

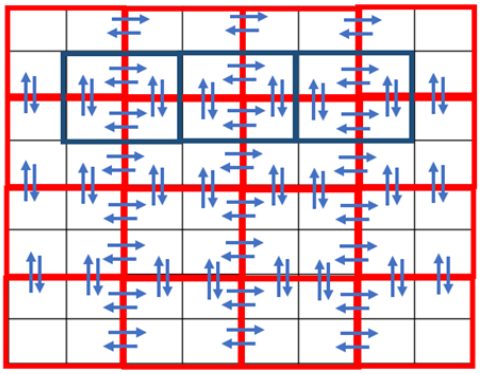


Fig 17: - Red Cells = Oddly placed pixels and this is where pooling happens

Blue = Evenly placed pixels and here the pooling does not happen

Arrows = Show the exchange of information.

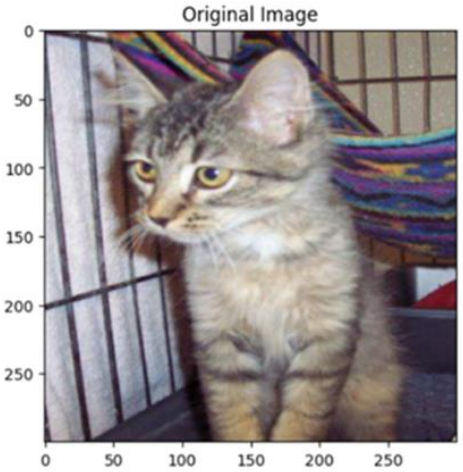


Fig 18: - Image before applying Texture Information Encoding augmentation

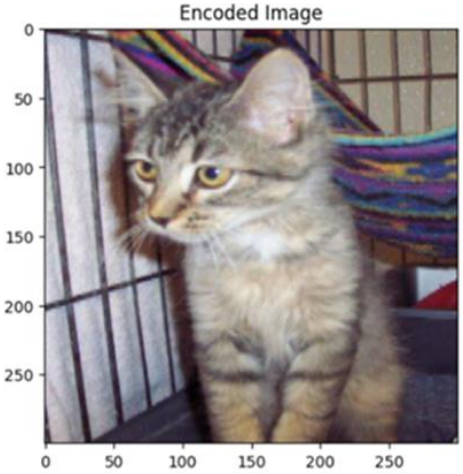


Fig 19: - Image after applying the Texture Information Encoding augmentation

TABLE II

Channel	Structural Similarity Score
Red	0.9677
Green	0.9679
Blue	0.9670

Channel-wise comparison between the original image and texture information encoded image.

We can see that in all channels there is 4% that is not the same. Here, that 4% is nothing but the texture information that is incorporated.

The images will look identical as that is the goal; we don't want to change the image completely; rather, we want to include the required information without much change in the original image.

Advantages:

- Using this technique, we can incorporate the required texture information without producing many changes.
- This is an information-preserving technique, so during the pooling operation, even though the window does not slide over evenly spaced pixels, we can still get the same impact as if it was sliding over evenly spaced pixels
- For applications that deal with texture information, this technique can have a good impact.

Disadvantages:

- As images are going to be identical, in terms of structural detection, it may not improve the model's performance.
- If texture information is not the main goal, then this technique may not be appropriate.

Transferable Model

A Transferable Model is a type of model where the knowledge learned from one part of the model can be shared with another part of the model. In other words, it is an inter-model communication mechanism. In this approach, we train the model to learn the required features, using the augmentations; we ensure that the features learned are robust in terms of representation, and after that, we transfer the weights from one part of the layer to another with the required transformation. Due to that, we need not provide every type of data (if the knowledge is transferable). So, rather than training the model extensively, we only need to train the model at the required amount and the rest of it can be simply transferred. Note: This approach works only if the information is transferable. For example, information on the features learned from a 0° image can be transformed and applied to the same image rotated at 270°. That means information learned at 0° can be transferred to various layers dealing with particular orientations. In other words, it is about inter-model communication mechanisms. In our brain, we have this internal communication where information learned in one area can be used in another area. In the same way, if we can develop models with these capabilities, we can make robust models with less training data and in smaller sizes.

As these models are not required to learn much information, we can reduce the number of filters, which in turn results in models with fewer memory requirements. As these models have fewer training parameters, they can be trained faster. In domains like virtual assistants, self-driving, etc. we use these types of models as they are smaller in size and information can be transferred and learned in real-time. If thorough research is put into this idea, we can correlate learning with the transfer of knowledge. Both terms can be used interchangeably.

Achieving Color Invariant Model

Here, the idea is that during the training process, the model learns weights according to RGB order (refer to Fig. 19). That means, the model learns the kernels according to the red color channel, Green channel, etc. Now, we will transfer the information (kernel) from one channel to another. That is, the information learned by R-channel will be shared with Blue and Green, and so on (refer to Fig. 20). Due to this, we need not train the model with color-augmented data. This is done by generating different combinations of RGB channels.

Steps:

1. Train the trainable layer (center) using robust augmentation techniques to improve its feature detection capabilities (refer to Fig. 21).

2. Transfer the weights of each associated channel in the required order of RGB to the appropriate layer (refer to Fig. 22).

Combinations: RGB, RBG, BGR, BRG, GRB, GBR.

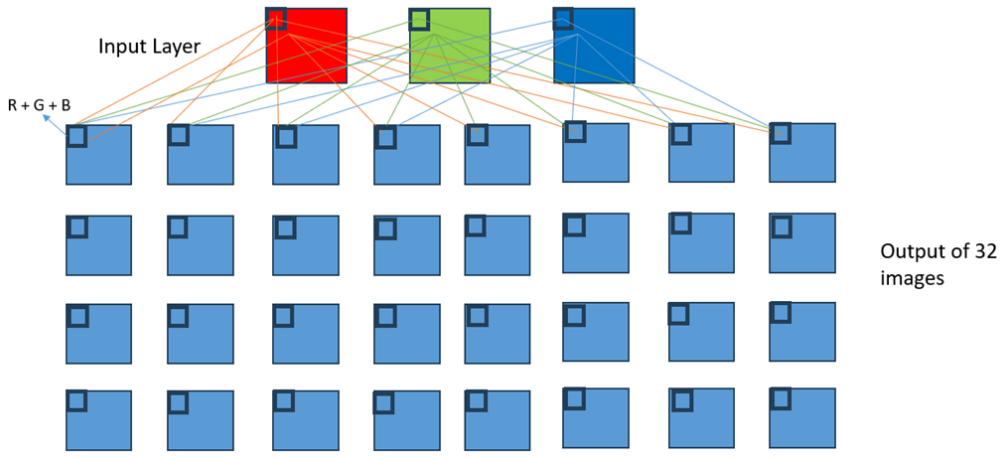


Fig 20: - Representation of the convolution operation at the first layer with the weights and connections associated with each channel

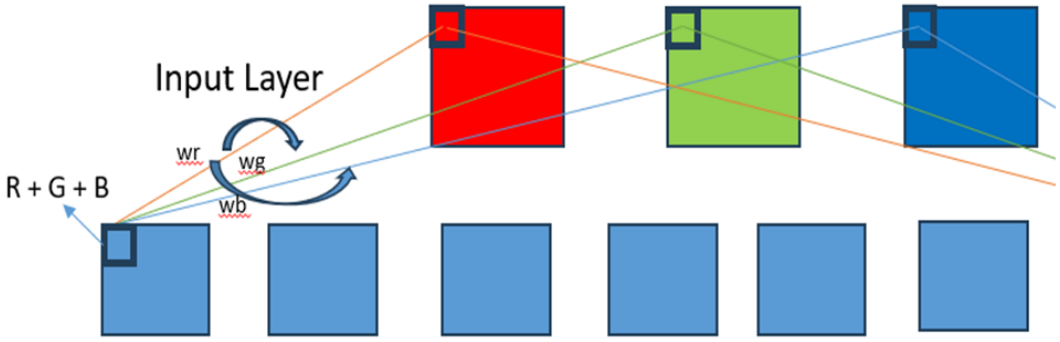


Fig 21: - Showing how we can transfer the knowledge from the Red to Green and Blue color channels.

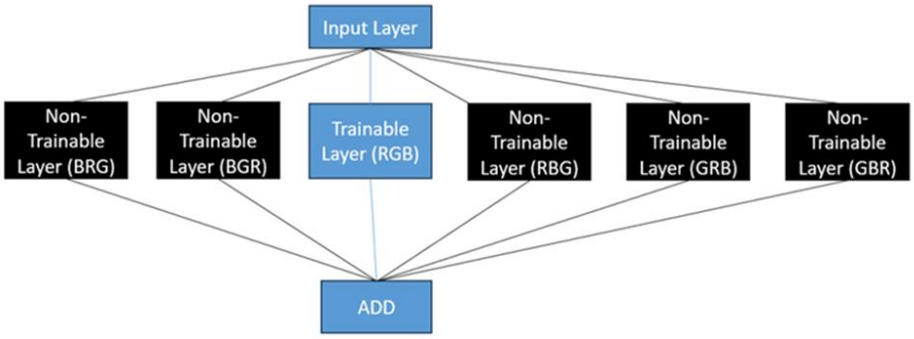


Fig 22: - Only the center layer is trained and the rest of the layers are initialized with zero weights and marked non-trainable.

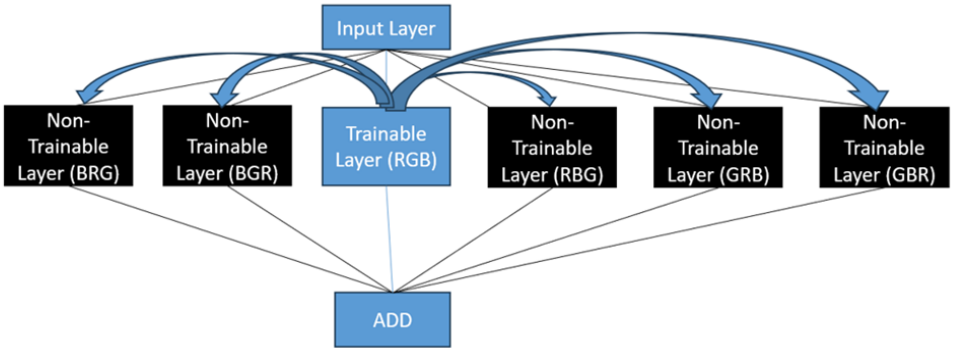


Fig 23: - After training, the weights of RGB channels are shuffled into different R-G-B combinations like BGR, BRG, etc., and initialized to the appropriate layer

Achieve Rotational Invariant Model

Here, the idea is that during the training process, the model learns weights to detect the features. After learning the kernel weights, the weights are transferred to other layers, which maintain information about 90, 180, and 270 angles. After passing the data into each of the layers, the kernels that are activated with the maximum value are selected.

Steps:
1. Train the trainable layer (center) using robust augmentation techniques to improve its feature representation (refer to Fig. 23).
2. Transfer the weights learned from the trainable layer to the appropriate non-trainable layer with the required transformation (refer to Fig. 24).

Reasons to choose only [90, 180, 270]:

• These are the angles that have the highest variance. So, the more variance in the training set the more information the model has to learn.

• The deeper layer is sensitive to rotations. So, if we have these rotational invariant blocks from the top layers, as we go deeper, the data settles down at these angles.

• Also, to ensure that the model is not very wide. The model which is very wide has high memory requirements as well as it takes more time to perform predictions.

• Rotating kernels at these particular angles will not introduce any new weights due to the interpolation.

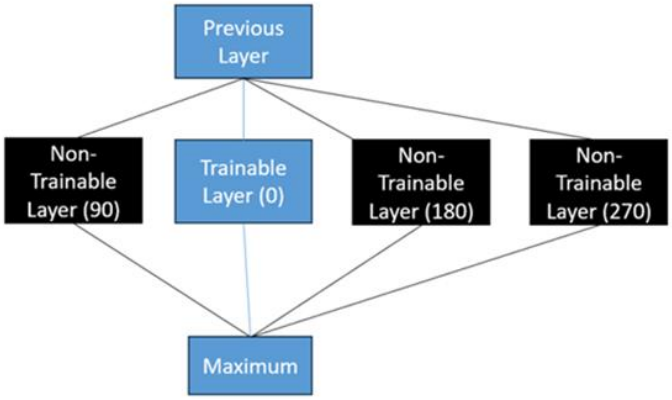


Fig 24: - Only the center layer is trained and the rest of the layers are initialized with zero weights and marked non-trainable.

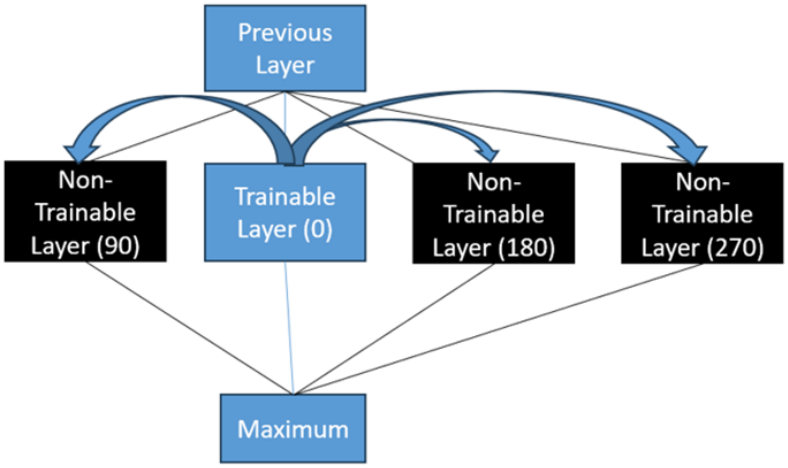


Fig 25: - The information from the trainable layer is transformed as required and shared.

Advantages:

- This design is inspired by the workings of the brain. It is an efficient way of learning.
- This approach can have more impact with less data and training time as we just need to transfer the rest of the knowledge.
- As we have fewer training parameters, we can reduce the size of the model.
- These models can change their learning in real time. Due to this, these types of models can be used for mobile phones, etc.

Disadvantages:

- We can get this advantage only if the knowledge is transferable.
- The design can be complicated.
- As we make the model wider, the prediction time can increase.
- In this approach, as we are manipulating the kernel weights, it will damage the inference mechanism of the model. To solve this, we need to retune the model by unfreezing all the layers.

IV. EXPERIMENT ANALYSIS

1. 1- Pixel Shift + Flip: -

Training Conditions:
Original Data: 1500 Cats, 1500 Dogs, 1500 NT
Augmented Data (1p-Flip): 1500 Cats, 1500 Dogs, 1500 NT
Total: 3000 Cats, 3000 Dogs, 3000 Nothing

Testing Conditions:
Test Dataset: 11000 Cats, 11000 Dogs

Note: NT stands for nothing, it is an extra class generated using Tile Shuffling augmentation to convert the binary classification to categorical classification.

TABLE II

Model	Cat Accuracy	Dog Accuracy	Net Accuracy
MODEL A1 (Flip Without 1 Pixel Shift)	89.19%	79.34%	84.26%
MODEL A2 (Flip with 1 Pixel Shift)	**86.21%**	**86.41%**	**86.31%**
MODEL B1 (Flip Without 1 Pixel Shift)	93.13%	83.29%	88.21%
MODEL B2 (Flip with 1 Pixel Shift)	**88.26%**	**88.13%**	**88.19%**

Improvement of results in comparison with model A1 which is trained on flip without 1 pixel shift and model A2 which is trained on flip with 1 pixel shift. And the same goes for B1 and B2 models.

2. Activation Suppressor:

Training Conditions**:**

Original Data: 3000 Cats, 3000 Dogs, 3000 NT
Augmented Data (AS*): 3000 Cats, 3000 Dogs, 3000 NT
Total: 6000 Cats, 6000 Dogs, 6000 Nothing

Testing Condition**:**

Test Dataset: 10740 Cats, 10740 Dogs

Note: AS* = Activation Suppression

TABLE III

Model	Cat Accuracy	Dog Accuracy	Net Accuracy
MODEL A1 (Without active suppression) (Base Model)	94.14%	78.64%	86.4%
MODEL A2 (With active suppression)	**88.33%**	**86.41%**	**88.71%**
MODEL A3 (With active suppression + effect)	**91.21%**	**92.82%**	**92.015%**
MODEL B1 (Without active suppression) (Base Model)	74.69%	85.2%	79.94%
MODEL B2 (With active suppression)	**88.82%**	**87.44%**	**88.13%**

In this table, model A1 is without active suppression, and after applying active suppression we got model A2 which improved its accuracy, and model A3 with active suppression and style effect model A3 has achieved 90-92% accuracy same pattern with the second model (B) also.

3. Adaptive Color Scheme:

Training Conditions:

Original Data: 1500 Cats, 1500 Dogs, 1500 NT
Augmented Data (ACS* applied): 1500 Cats, 1500 Dogs, 1500 NT
Total: 3000 Cats, 3000 Dogs, 3000 Nothing

Testing Conditions:

Test Dataset: 3000 Cats, 3000 Dogs
It is the same as the training dataset but here on each image, a random color is applied so that we can test the model's robustness to colors.
Note: ACS* = Adaptive Color Scheme

TABLE IV

Model	Cat Accuracy	Dog Accuracy
Base Model	75.03%	56.86%
Model Trained on data generated by adaptive color scheme	97.56%	84.53%

Improvement of results in comparison with the Base Model and the Model trained on data generated by adaptive color scheme.

4. Tile shuffler:

Training Conditions:

Original Data: 1500 Cats, 1500 Dogs, 1500 NT

Testing Conditions:

Test Dataset: 10740 Cats, 10740 Dogs

TABLE V

Model	Cat Accuracy	Dog Accuracy	Net Accuracy
MODEL A1 (binary)	75.1%	74.55%	75.32%
MODEL A2 (categorical)	**81.56%**	**85.2%**	**83.38%**

Improvement of results in comparison with Model A1 which contains binary classes and Model A2 which contains categorical classes.

5. Low-Level Residuals: -

Training Conditions:

Original Data: 3000 Cats, 3000 Dogs, 3000 NT
Augmented Data (LLR*): 3000 Cats, 3000 Dogs, 3000 NT
Total: 6000 Cats, 6000 Dogs, 6000 NT

Testing Conditions:

Test 1: 10740 Cats, 10740 Dogs
Test 2: 3000 Cats, and 3000 Dogs of the training dataset where each image is randomly shifted between -50 to +50 pixels in both horizontal and vertical directions.
Note: LLR* = Low Level Residual

TABLE VI

Model	Cat Accuracy	Dog Accuracy
Test 1:		
Base Model	85.13%	83.29%
Model trained with LLR	90.74%	85.25%
Test 2:		
Base Model	92.30%	77.33%
Model trained with LLR	90.74%	85.25%

Models trained with LLR resulted in being more robust to the translation of features and also improved overall accuracy.

6. Texture Information Encoder: -

Training Conditions:
Original Data: 3000 Cats, 3000 Dogs, 3000 NT
Augmented Data (TIE*): 3000 Cats, 3000 Dogs, 3000 NT
Total: 6000 Cats, 6000 Dogs, 6000 NT

Testing Conditions:
Test Dataset: 10740 Cats, 10740 Dogs
Note: TIE* = Texture Information Encoder

TABLE VII

Model	Cat Accuracy	Dog Accuracy
Model - A1 Trained on Original Data	92.30%	77.33%
Model - A2 Trained on encoded images	96.76%	89.76%
Model - B1 Trained on Original Data	96.36%	93.80%
Model - B2 Trained on encoded Data	98.90%	95.37%

The models which are trained using TIE* resulted in improved overall accuracy. By encoding the information, without expanding the training dataset, we can get improved performance.

7. Tests on Transferable Model: -
Color Invariance Test:

Training conditions:
Original Data: 1500 Cats, 1500 Dogs, 1500 NT
Applied Augmentations: 1-Pixel Shift + Flip, Activation Suppression.

Test conditions:
Test Data: 3000 Cats, and 3000 Dogs from the training dataset with and without flipping with random shades stacked on top of them so that we can test the model's robustness to the color variations.

TABLE VIII

MODEL	RESULT
TEST1	
Model without any transfer of weights	Cat accuracy: 74.36% Dog accuracy: 61.83%
Model with a transfer of weights	Cat accuracy: 96.86% (**22.5+**) Dog accuracy: 86.76% (**24.9+**)
TEST2	
Model without any transfer of weights	Cat accuracy: 77.03% Dog accuracy: 51.03%
Model with a transfer of weights	Cat accuracy: 92.33% (**15.3+**) Dog accuracy: 80.60% (**29.5+**)
Note: - Amount of time taken to get this effective: under 5 sec.	

Rotational Invariance Test (Large Model – 10 MB):

Training Conditions:
Original Data: 1500 Cats, 1500 Dogs, 1500 NT
Augmentations Applied: 1-Pixel Shift + Flip, Activation Suppression, Negative Coloring, Bright and Dark Images.

Testing Conditions:
Test Dataset: 10740 Cats, 10740 Dogs with each image rotated at degrees mentioned in the table.

TABLE IX

Angle	Transferred Model	Non-Transferred Model
0º	**Cat accuracy: 90.12** **Dog accuracy: 89.72**	Cat accuracy: 90.22 Dog accuracy: 90.11
90º	**Cat accuracy: 88.21** **Dog accuracy: 87.69**	Cat accuracy: 75.18 Dog accuracy: 72.70
180º	**Cat accuracy: 86.75** **Dog accuracy: 88.57**	Cat accuracy: 64.40 Dog accuracy: 72.83
270º	**Cat accuracy: 88.00** **Dog accuracy: 87.34**	Cat accuracy: 76.28 Dog accuracy: 74.72
Random	**Cat accuracy: 88.39** **Dog accuracy: 89.23**	Cat accuracy: 83.06 Dog accuracy: 62.24

Note: Random means each image in the testing dataset is rotated at a random degree between 0 and 360.

By the above results, we can infer that no matter at what degree the images are being rotated, the accuracy is consistent in the transferable model and it is inconsistent in the non-transferable model.

Rotational Invariance Test (Small Model – 2.29 MB):
Training Conditions:
Original Data: 1500 Cats, 1500 Dogs, 1500 NT
Augmentations Applied: 1-Pixel Shift + Flip, Activation Suppression, Negative Coloring, Bright and Dark Images.

Testing Conditions:
Test Dataset: 10740 Cats, 10740 Dogs with each image rotated at degrees mentioned in the table.

TABLE X

Angle	Transferred Model	Non-Transferred Model
0º	**Cat accuracy: 90.12** **Dog accuracy: 89.72**	Cat accuracy: 90.22 Dog accuracy: 90.11
90º	**Cat accuracy: 88.21** **Dog accuracy: 87.69**	Cat accuracy: 75.18 Dog accuracy: 72.70
180º	**Cat accuracy: 86.75** **Dog accuracy: 88.57**	Cat accuracy: 64.40 Dog accuracy: 72.83
270º	**Cat accuracy: 88.00** **Dog accuracy: 87.34**	Cat accuracy: 76.28 Dog accuracy: 74.72
Random	**Cat accuracy: 88.39** **Dog accuracy: 89.23**	Cat accuracy: 83.06 Dog accuracy: 62.24

V. Conclusion

We have presented six new augmentation techniques (1-pixel shift + flip, Activation Suppression, Adaptive Color Scheme, Tiles Shuffling, Low-Level Residual, and Texture Information Encoder) some of which are adaptive based on the current ability of the model, and some of them are designed to address the drawbacks of the current CNN models. The techniques which we have presented are efficient in terms of the impact they are producing on the model. We have demonstrated the improvement of the model using these techniques in various areas. And also, we have presented a new idea called the Transferable model which the model will have the capability of performing intermodal communication. We have demonstrated how robust these models are to the variations in colors as well as rotations. We have shown how efficient these models are in terms of memory as well as the amount of training time required. If we can put more effort into these kinds of ideas where communication is similar to human brains, we can develop powerful deep learning models with no wastage of computation.

VI. References

[1] Nanni, Loris, Stefano Ghidoni, and Sheryl Brahnam. "Handcrafted vs. non-handcrafted features for computer vision classification." Pattern recognition 71 (2017): 158-172.

[2] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).

[3] Bellet, Marie E., et al. "Human-level saccade detection performance using deep neural networks." Journal of neurophysiology 121.2 (2019): 646-661.

[4] De Man, Ruben, et al. "Comparison of deep learning and human observer performance for detection and characterization of simulated lesions." Journal of Medical Imaging 6.2 (2019): 025503-025503.

[5] Alzubaidi, Laith, et al. "Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions." Journal of big Data 8 (2021): 1-74.

[6] Varoquaux, Gaël, and Veronika Cheplygina. "Machine learning for medical imaging: methodological failures and recommendations for the future." NPJ digital medicine 5.1 (2022): 48.

[7] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. Pattern recognition, 77, 354-377.

[8] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems 25 (2012).

[9] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." Journal of big data 6.1 (2019): 1-48.

[10] Zhang, Richard. "Making convolutional networks shift-invariant again." International conference on machine learning. PMLR, 2019.

[11] Zhong, Zhun, et al. "Random erasing data augmentation." Proceedings of the AAAI conference on artificial intelligence. Vol. 34. No. 07. 2020.

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

[13] Yun, Sangdoo, et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." Proceedings of the IEEE/CVF international conference on computer vision. 2019.

[14] Hendrycks, Dan, et al. "Augmix: A simple data processing method to improve robustness and uncertainty." arXiv preprint arXiv:1912.02781 (2019).

[15] Walawalkar, Devesh, et al. "Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification." arXiv preprint arXiv:2003.13048 (2020).

[16] Patel, Omprakash, Yogendra PS Maravi, and Sanjeev Sharma. "A comparative study of histogram equalization based image enhancement techniques for brightness preservation and contrast enhancement." arXiv preprint arXiv:1311.4033 (2013).

[17] Sabour, Sara, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic routing between capsules." Advances in neural information processing systems 30 (2017).

[18] Jian, S., et al. "Deep residual learning for image recognition." IEEE Conference on Computer Vision & Pattern Recognition. 2016.

[19] Eren Akbiyik, M. "Data Augmentation in Training CNNs: Injecting Noise to Images." arXiv e-prints (2023): arXiv-2307.

[20] Laptev, Dmitry, et al. "Ti-pooling: transformation-invariant pooling for feature learning in convolutional neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.
[21] Worrall, Daniel E., et al. "Harmonic networks: Deep translation and rotation equivariance." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[22] Mo, Hanlin, and Guoying Zhao. "RIC-CNN: rotation-invariant coordinate convolutional neural network." arXiv preprint arXiv:2211.11812 (2022).
[23] Jaderberg, Max, Karen Simonyan, and Andrew Zisserman. "Spatial transformer networks." Advances in neural information processing systems 28 (2015).
[24] Simonyan, Karen, Andrea Vedaldi, and Andrew Zisserman. "Deep inside convolutional networks: Visualising image classification models and saliency maps." arXiv preprint arXiv:1312.6034 (2013).
[25] Maharana, Kiran, Surajit Mondal, and Bhushankumar Nemade. "A review: Data pre-processing and data augmentation techniques." Global Transitions Proceedings 3.1 (2022): 91-99.
[26] Khalifa, Nour Eldeen, Mohamed Loey, and Seyedali Mirjalili. "A comprehensive survey of recent trends in deep learning for digital images augmentation." Artificial Intelligence Review (2022): 1-27.
[27] Xu, Mingle, et al. "A comprehensive survey of image augmentation techniques for deep learning." Pattern Recognition (2023): 109347.