Learning Perturbations To Improve Classification Accuracy

A Report Submitted
in Partial Fulfillment of the Requirements for the Course

A.I. Specialization Project (CS496)

by

Qazi Sajid Azam
(B16CS026)

Anurag Shah
(B16CS034)

under the guidance of

Dr. Deepak Mishra

to the

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY JODHPUR

CERTIFICATE

*This is to certify that the work contained in this thesis entitled "**Learning Perturbations To Improve Classification Accuracy**" is a bonafide work of **Qazi Sajid Azam** (**Roll No. B16CS026**) and **Anurag Shah** (**Roll No. B16CS034**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Jodhpur under my supervision and that it has not been submitted elsewhere for a degree.*

Supervisor: **Dr. Deepak Mishra**

Assistant Professor,

Department of Computer Science & Engineering,

Indian Institute of Technology Jodhpur, Rajasthan.

July, 2020

Jodhpur.

Acknowledgements

We would like to thank **Dr. Deepak Mishra** for guiding us during the course of this project.
We also acknowledge the institute for providing the necessary platform and resources for completing this project.

Contents

List of Figures

Chapter 1

Introduction

In recent years, Convolutional Neural Networks have been used extensively for a variety of tasks. While we are happy with the performance of these systems, it is also a very interesting question to consider how exactly a CNN works.

In our project we focus on an image classification model based on VGG-16 [6]. We want to get an insight about what characteristics in the image influence the decision of the classifier. In other words, we want to know which characteristics of the image were major factors in the classification process.

We explore numerous experiments, and in the end we conclusively show that modifying certain things in the image leads to better accuracy in classification, while providing a reasonable explanation of why it works like this.

Chapter 2

Literature Survey

2.1 LIME - Local Interpretable Model-Agnostic Explanations

LIME [2] is an explanation technique that explains the predictions of any classifier in an interpretable and faithful manner, by learning an interpretable model locally around the prediction.

2.2 GRADCAM - Gradient-weighted Class Activation Mapping

In this paper [5], the authors propose a method to identify important regions of the image that are being focused on by a classifier CNN. It uses the gradients of any target concept, flowing into the final convolutional layer to produce a coarse localization map highlighting important regions in the image for predicting the concept.

2.3 Uncovering the impact of Background Features on Deep Neural Networks

In this paper [4], the authors explore the impact of image background in the classification tasks. They demonstrate that in many cases, a classifier which seems to have learned to identify the foreground object, is actually relying on the background information.

2.4 Transferable Recognition-Aware Image Processing

In this paper, [1], the authors tell that when we use any image transformation network, such as denoiser, the output does not look like a different object to the human, but many classifier networks give wrong output for these images. To address this problem, they use a loss component based on classification and look for a solution for transferable network among different classifiers.

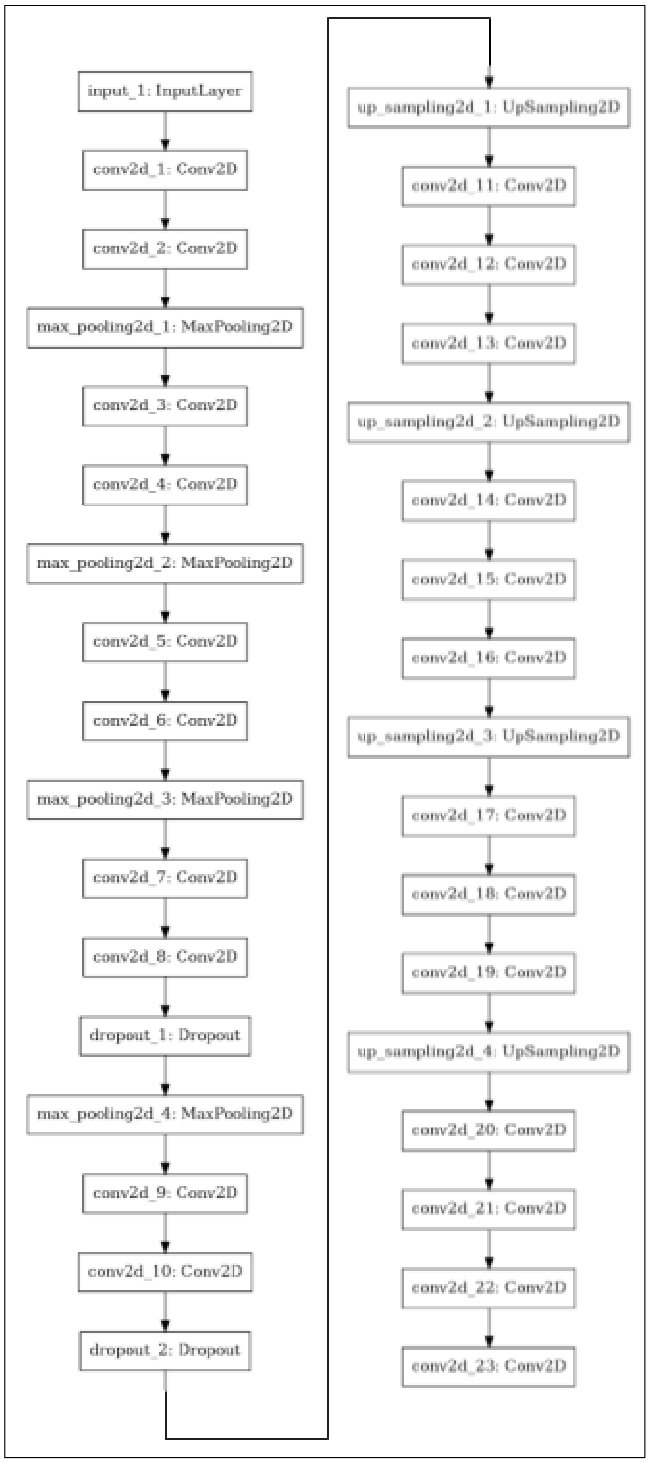
Chapter 3

Methodology and Work Done

We first train the classifier on the selected dataset till we achieve a certain accuracy. Then we train an image transformation network, which is simply an autoencoder designed to reproduce the input image. The initial training of autoencoder is done using conventional MSE loss. For the next part, we fine-tune the autoencoder using a cross-entropy loss based on the accuracy of classifier upon the output of the autoencoder. It is important to note here that while finetuning the autoencoder, the classifier weights are frozen.
The expectation is that the autoencoder will learn to modify the input image in such a way that the final classification accuracy is improved. This will give us some information about what characteristics the classifier is looking for in an image.
We further use the GRADCAM model to generate attention maps, which we can use to verify the above approach.

3.1 Network Architecture

For the classifier, we use an architecture based on VGG-16 [6]. The autoencoder is based on the U-net architecture [3]. We have made some minor modifications in both models according to our dataset. The complete training procedure is shown in figure 3.4. The individual architectures are shown in figure 3.1.



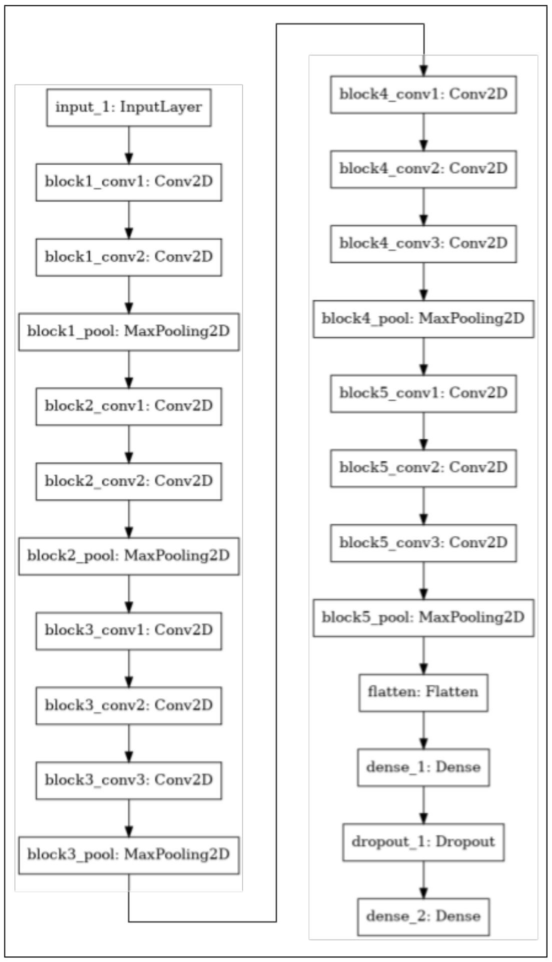


Fig. 3.1 Network architecture of autoencoder (left) and classifier (right)

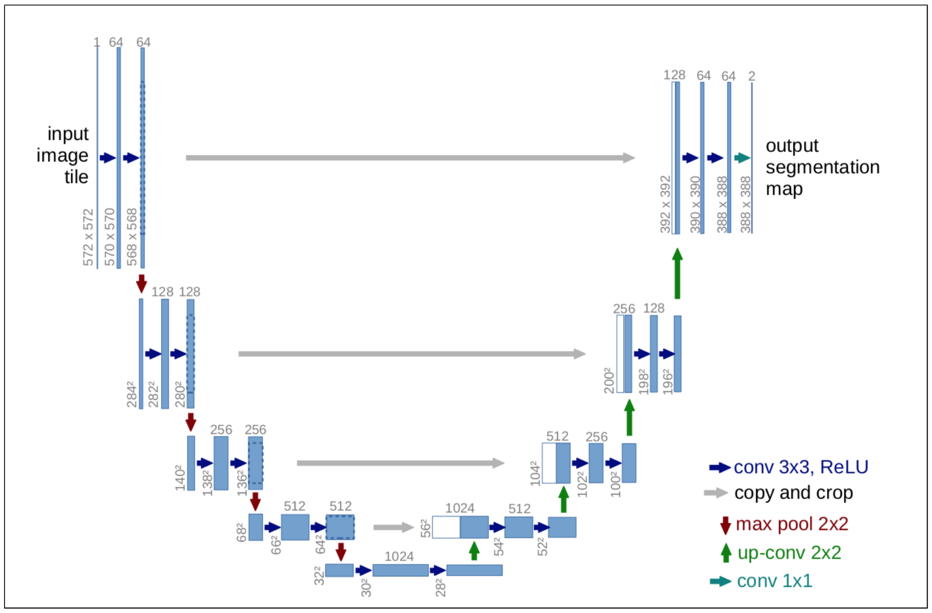


Fig. 3.2 UNet Model as in original paper

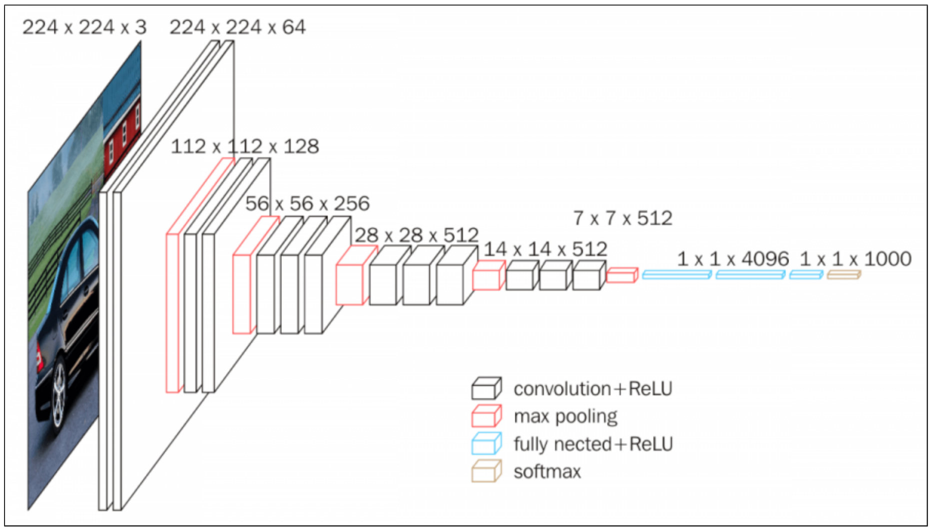


Fig. 3.3 VGG16 Model as in original paper

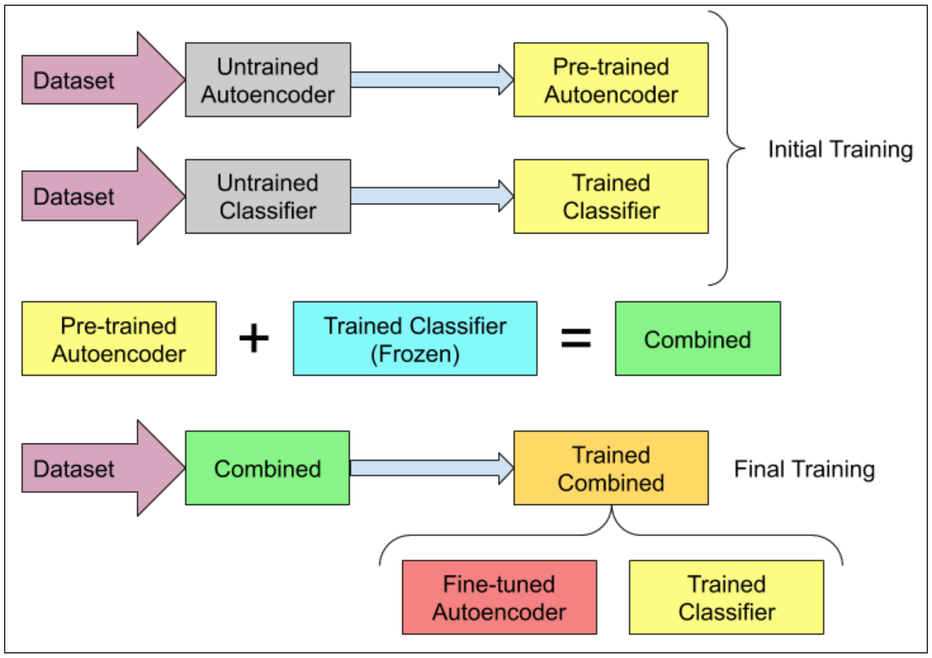


Fig. 3.4 Block diagram of the training process

3.2 Dataset

For the various experiments, we mainly use the CIFAR-10 dataset and STL-10 dataset, and also one dataset from Kaggle, called Intel Image Classification dataset. The CIFAR-10 dataset contains 50000 training images and 10000 testing images, and images are divided into 10 classes. The STL-10 dataset has higher resolution images 96×96 compared to CIFAR-10, which has 32×32. It contains 5000 images divided into 10 classes. We manually divide it into 4010 images for training and 990 images for testing. The Intel dataset contains 14000 images of 150×150 size, and we divide these into 10000 images for training and 4000 for testing.

3.3 Loss Functions

The initial training of autoencoder uses Binary Cross Entropy loss shown in equation 3.1, in order to learn to reproduce the image. It is a Sigmoid activation plus a Cross-Entropy loss. BCE is also known as Sigmoid Cross-Entropy loss. It is independent for each vector component (class), meaning that the loss computed for every CNN output vector component is not affected by other component values. That's why it is used for multi-label classification, were the insight of an element belonging to a certain class should not influence the decision for another class. It's called Binary Cross-Entropy Loss because it sets up a binary classification problem between C'=2 classes for every class in C. Intuitively why it is appropriate for our usecase is that for each pixel in the output of the UNet model this loss tells us how much correlated it is to the corresponding pixel in the input image. Our aim is to get maximum correlation with the original input image.

$$BCE = -\sum_{i=1}^{C'=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1)) \tag{3.1}$$

The training of classifier uses cross-entropy loss according to equation 3.2. The fine-tuning phase of the autoencoder training also uses the same categorical cross-entropy loss of classifier as its own loss, while the classifier remains fixed.

$$CCE = -log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j} a}\right) \tag{3.2}$$

Chapter 4

Experiments and Results

As discussed before, the key components in our project are the autoencoder, classifier and the dataset. We begin with a very simple autoencoder with 1 encoding and 1 decoding layer, and a generic CNN classifier. We used the CIFAR-10 dataset for this. This experiment did not yield good results, so we discarded it. For the next experiment we improve the autoencoder to 4 layers each for encoding and decoding, replaced the classifier with pre-trained VGG-16, and used the STL-10 dataset, but still did not get desired results. After this, we modified the VGG-16 by adding two dense layers at the end, and did fine-tuning of the the pretrained VGG using these layers. This gives us slightly better results but not good enough. From here onwards we use the U-net architecture for the autoencoder, and use binary cross-entropy loss. This approach gives us much better results, and we tried variations of different parameters such as number of epochs, to control the accuracy of classifier. All the further results support our initial claim that modification of image can improve classification accuracy.

4.1 Image Differences

For analyzing the action of autoencoder, we generate the difference images by subtracting the input image from output and taking absolute values. Then we perform contrast stretching on the difference, and apply heatmap upon it for better visual clarity. It can be seen clearly that autoencoder is enhancing the key details of image such as edges and corners. It also improves the color contrast. All these are changes which we expect to improve accuracy even from human perspective. We expect that a sharp and bright image is easier to identify than a dull and blurry image.

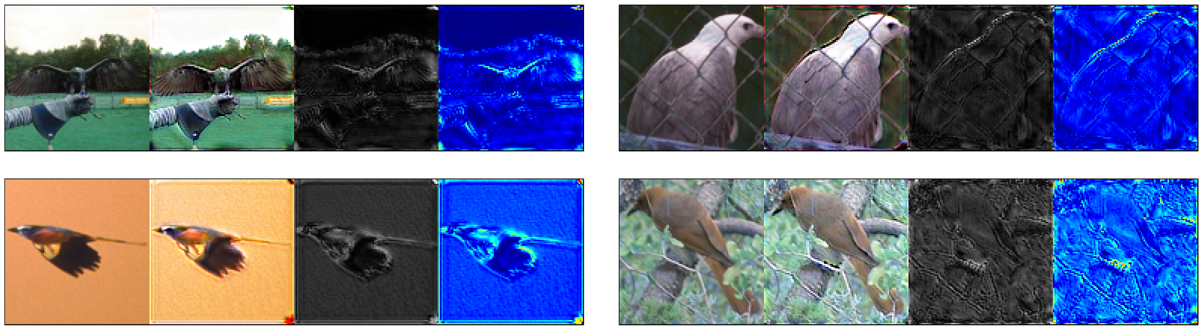


Fig. 4.1 Image differences with heatmap visualization. From left to right the images correspond to the original image, Autoencoder output, Difference between original image and autoencoder output, heatmap of the difference.

4.2 Attention Heatmaps

We use GRADCAM [5] to identify which regions of the image the classifier is looking at. In many cases, we see that when original image is put into the classifier, it is looking at irrelevant parts, which might cause wrong output. By passing the original image through the autoencoder, we get the output image. When this image is put into the classifier, it now looks at the relevant regions and thus gives better results.

As discussed in [4], classifier might learn a wrong correlation between object class and the background, making it give wrong results. Using the attention maps, we can see that first it was looking at background region which causes wrong output. For example, the sky background is common in airplane images, which creates a bias, thus classifying another image with sky background as airplane without looking at the foreground object. It may classify a bird flying in the sky as airplane.

After the combined training, we see that there is huge improvement and it correctly focuses on the foreground object, resulting in accurate classification.

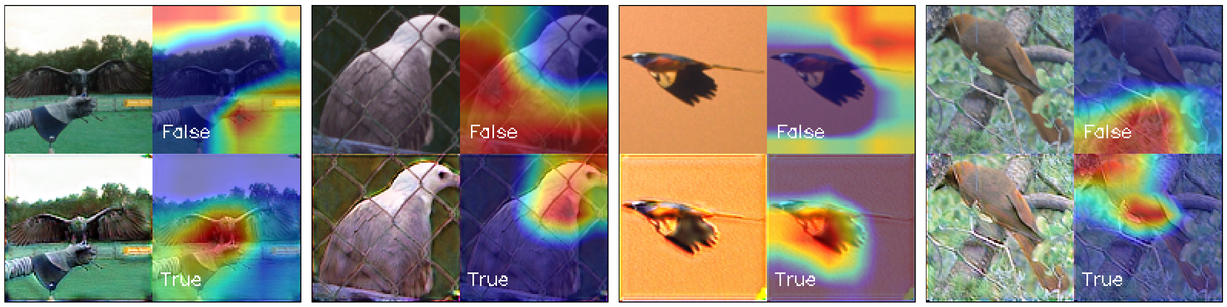


Fig. 4.2 Attention Heatmap Comparisons. From top left moving in clockwise direction the images correspond to original image, GRADCAM of classifier(VGG16), GRADCAM of our model, autoencoder output. True and False correspond to whether the classfication was correct or not.

4.3 Network Internal Representations

We extract the internal representations of the classifier when it is working on the given image, in order to analyze in more depth about what it is doing. In figure 4.3, we show the representation of image at first and second convolutional layer, in the first and second rows respectively. In each row, the left side image corresponds to the original image, and the right side corresponds to transformed image. On close observation, the right images are more finely detailed than the left side.

4.4 Quantitative Analysis

The table 4.1 shows detailed quantitative results of the several experiments we have performed. The most important thing to note is that accuracy is increased. In the table, 'own(a-b)' signifies that we used own autoencoder with 'a' encoding layers and 'b' decoding layers. We have performed more experiments than what is included in the table, but they are redundant as far as the conclusion is concerned. We have included the key experiments that correspond to some significant observation.

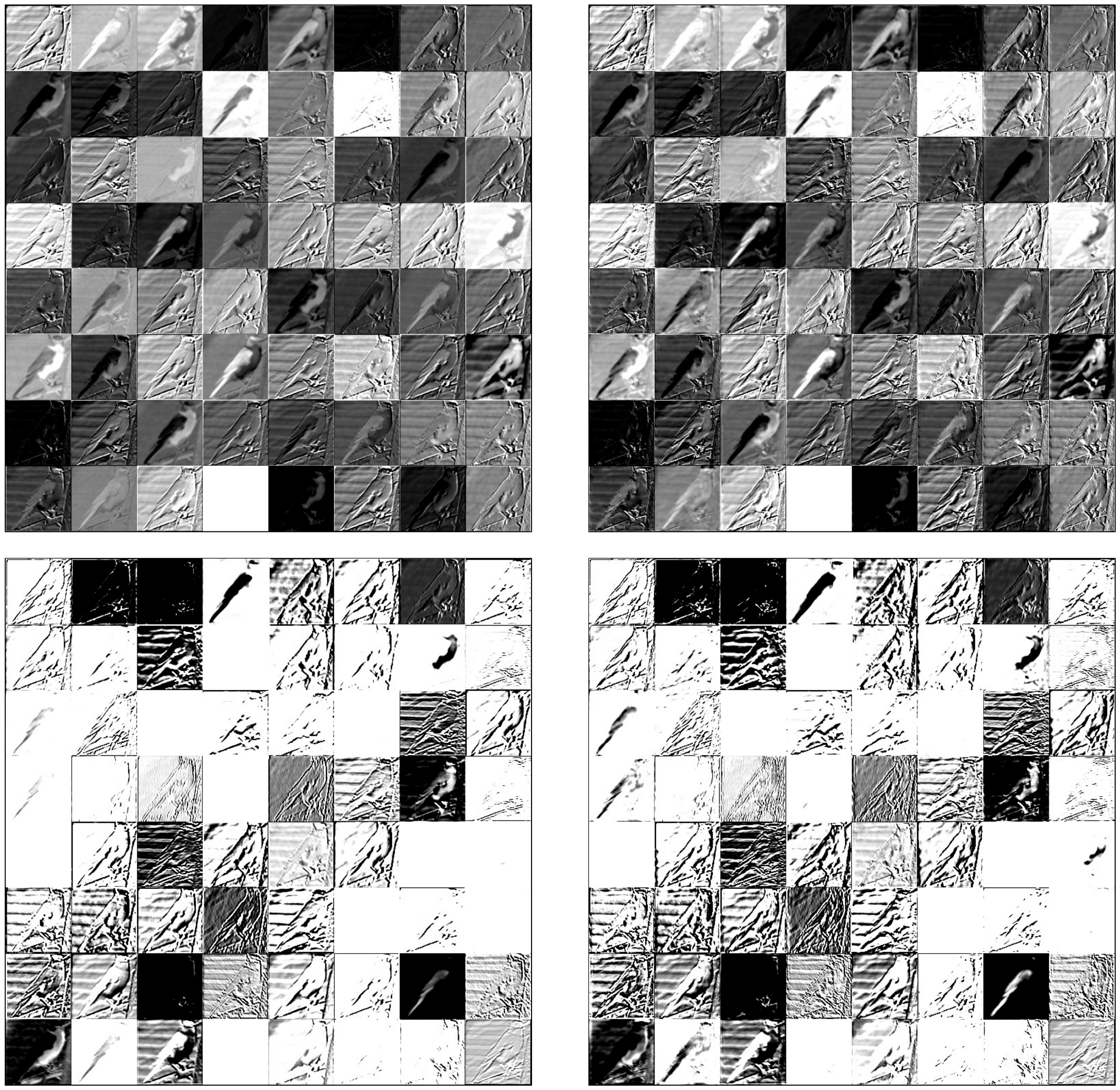


Fig. 4.3 Network internal representations

S.No	Dataset	Autoencoder	Classifier	Classifier Acc.		Combined Acc.	
				Train	Test	Train	Test
1	STL-10	own (4-4)	VGG16	94.2	81.9	99.1	66.7
2	STL-10	own (4-4)	VGG16	100	81.9	100	66
3	STL-10 (full)	U-net	VGG16	99.6	82.8	99.7	96.0
4	STL-10	U-net	VGG16	99.6	82.8	99.2	85
5	STL-10 (full)	U-net	VGG16	89.4	82.6	99.7	98.7
6	STL-10	U-net	VGG16	89.4	78.5	99.0	82.3
7	STL-10	U-net	VGG16	99.6	82.6	100	84.3
8*	CIFAR-10	U-net	VGG16	99.6	75.7	99.8	79.6
9*	INTEL-6	U-net	VGG16	99.9	90.6	99.5	93.2
10*	INTEL-6 (full)	U-net	VGG16	99.9	90.6	99.5	99.0

Table 4.1 Quantitative results of the key experiments
* - The images from these datasets have not been analysed, only accuracy is calculated

Chapter 5

Conclusions

We have shown how classifier despite giving good accuracy focuses on wrong part of image for classification. We further proposed a method to increase the accuracy of classifier and showed how our model focuses on the right part of image for classification. We performed experiments on 2 different datasets and achieved better accuracy than the standard classifier VGG16.

Our objective was to show that certain modifications in the image might improve the classification accuracy. From a logical perspective also, it seems that highlighting certain parts of the image will make it easier to identify the main object. Even for a human eye, a dull image might seem to be dry grass, but upon sharpening and improving colour contrast, we might notice that there is a tiger sitting in middle. A similar observation is made in our experiments. As shown in the figures previously, the processed images exhibit greater visual clarity compared to the original, which is a factor in the accuracy of classification.

References

[1] Zhuang Liu et al. *Transferable Recognition-Aware Image Processing*. 2019. arXiv: `1910.09185 [cs.CV]`.

[2] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin. “”Why Should I Trust You?”: Explaining the Predictions of Any Classifier”. In: *CoRR* abs/1602.04938 (2016). arXiv: `1602.04938`. URL: `http://arxiv.org/abs/1602.04938`.

[3] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *CoRR* abs/1505.04597 (2015). arXiv: `1505.04597`. URL: `http://arxiv.org/abs/1505.04597`.

[4] Vikash Sehwag et al. *Time for a Background Check! Uncovering the impact of Background Features on Deep Neural Networks*. 2020. arXiv: `2006.14077 [cs.CV]`.

[5] Ramprasaath R. Selvaraju et al. “Grad-CAM: Why did you say that? Visual Explanations from Deep Networks via Gradient-based Localization”. In: *CoRR* abs/1610.02391 (2016). arXiv: `1610.02391`. URL: `http://arxiv.org/abs/1610.02391`.

[6] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: (2014). arXiv: `1409.1556 [cs.CV]`.