Machine Learning Practicals: Final Report

G010 (s1800640, s1807428)

Abstract

This paper will investigate how AlexNet, a convolutional neural network (CNN) used for image classification (Krizhevsky et al., 2012), would behave if the kernels of first layers are replaced by pre-defined Gabor, Blob and Edge Filters. ~~By analysing the results of the experiments, we would claim~~ that deep net training incorporates random variables and that pre-defined kernels contribute little to the accuracy and speed of learning.

1. Introduction

The process of categorising and labelling groups of pixels or vectors within an image based on specific rules is known as image classification (ima, 2009). Image classification is a complex procedure that relies on a number of factors and can be a challenging task to accomplish at times (Sangaiah, 2019). Deep learning has sparked a lot of interest in recent years (Goodfellow, 2016). Convolutional neural network (CNN), a class of artificial neural networks that has been a dominant method in computer vision tasks since the astonishing results on object recognition tasks were shared, is the most established algorithm among various deep learning models (Russakovsky et al., 2014). Among the numerous modifications and extensions to the fundamental CNN design, one architecture called AlexNet achieved record-breaking results using only supervised learning on a particularly hard dataset (Krizhevsky et al., 2012).

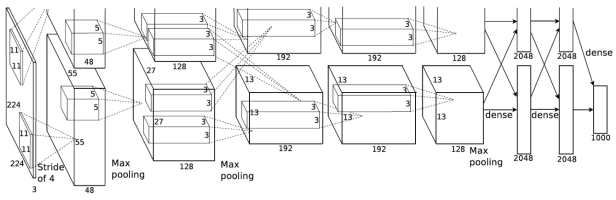

Figure 1. AlexNet Architecture designed by (Krizhevsky et al., 2012)

The architecture of AlexNet is depicted in Figure 1. When we examine the structure of the AlexNet, we find that the first convolutional layer of AlexNet filters the 224×224×3 input image with 64 kernels of size 11×11×3 with a stride of 4 pixels. In other words, given an image of size 224×224 with red, green and blue ~~three~~ color planes, a kernel of size 11×11×3 ~~convolve~~ the image input and output a result of size 55×55. The kernel of size 11×11×3 can be regarded a tiny image as well, which piqued our attention in the final appearance of the 64 kernels in the pre-trained AlexNet model.

We downloaded the pretrained AlexNet from PyTorch Hub (Pytorch_Team) and rendered the first layer as 96 normal RGB pictures as shown in Figure 2.

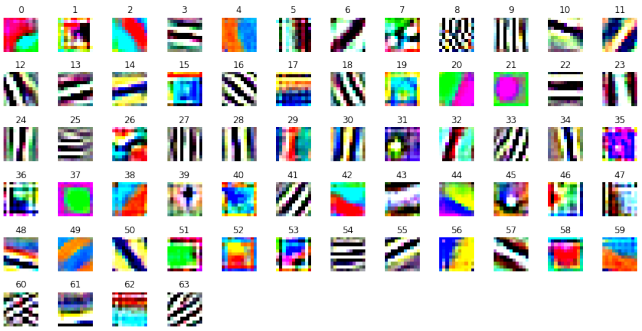

Figure 2. Visualisation of the kernels of first layer from AlexNet (Krizhevsky et al., 2012)

The visualisation of the first layer kernels from AlexNet is pretty surprising, since it appears that some kernels exhibit particular patterns. For example, kernels with indexes 3, 9, and 12 are composed of straight stripes of varying colours and angles. The kernels with indexes 20 and 59 are made up of two primary colours and have a distinct straight line boundary. The indexes 21 and 37 are shaped like a circle encircled by another colour. Meanwhile, other kernels, such as kernel 46 and 47, appear to be completely random.

To have a better understanding of the kernels, we extract all three layers from each kernel and represent them independently as shown in Figure 3.

We discover an intriguing phenomenon: some kernels have two colour planes with same shapes but roughly opposite values, whereas others have identical shapes and values.

The forms of the kernels in the first layer of AlexNet remind us of several current filters in the field of computer vision, including Gabor Filters, Blob Filters, and Edge Filters. In Image processing, a Gabor filter, named after Dennis Gabor, is a linear filter used for texture analysis, which entails determining whether an image contains any specific frequency content in specific directions within a confined region surrounding the point or region of investigation (Aach et al., 1995). A blob filter (also known as Particle Filter or Analysis) enables the identification of a specific blob based on

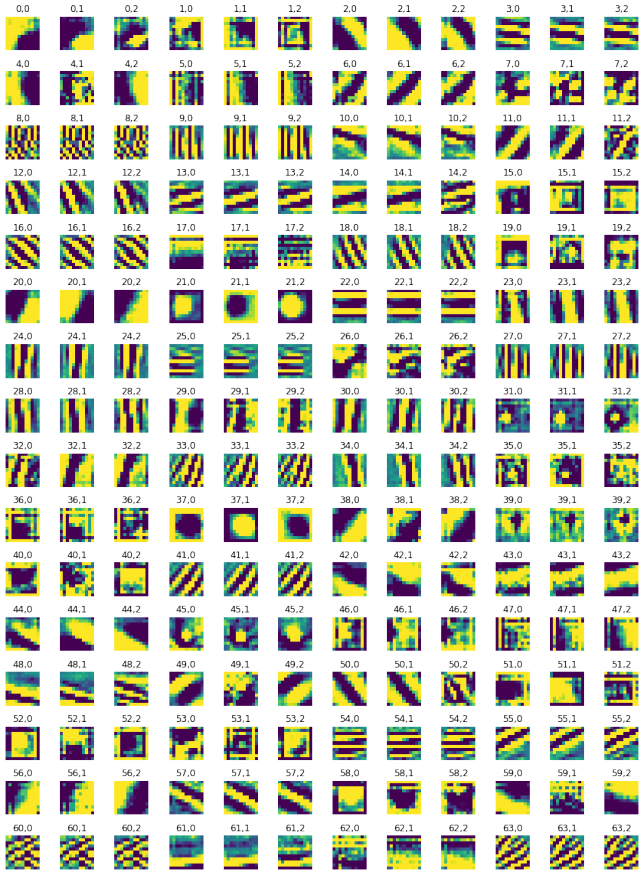

Figure 3. Visualisation of the kernels of first layer in single channel from AlexNet (Krizhevsky et al., 2012)

attributes that characterise the blob's properties and relationships to other blobs (Gustafsson, 2010). An Edge filter divides a broad spectrum of light into two components: one that is transmitted and one that is blocked (Snyder & Fedorovskaya, 2011).

Objective: We observed consistent patterns in the first layer of a pre-trained AlexNet model. These patterns are associated with some predefined vision filters. As such, we desire to investigate how AlexNet would behave if the first layer's kernels were replaced by pre-defined Gabor, Blob, and Edge Filters.

Hypothesis: We propose that by replacing or altering the first layer of **OriginalAlexNet**, the model will converge faster while maintaining nearly the same model prediction accuracy performance. Meanwhile, by replacing or altering the first layer of **PretrainedAlexNet**, the model could have better prediction accuracy performance.

Dataset: The dataset on which we will base our analysis is retrieved from Kaggle (Banerje). The collection contains 5400 images of animals classified into 90 distinct classifications, which should be sufficiently challenging for a deep learning classifier to learn so that we can determine whether any distinction exists for different Deep Net models. The example figures of 9 classes among 90 classes are shown in Figure 4. Apart from the colour photos, we

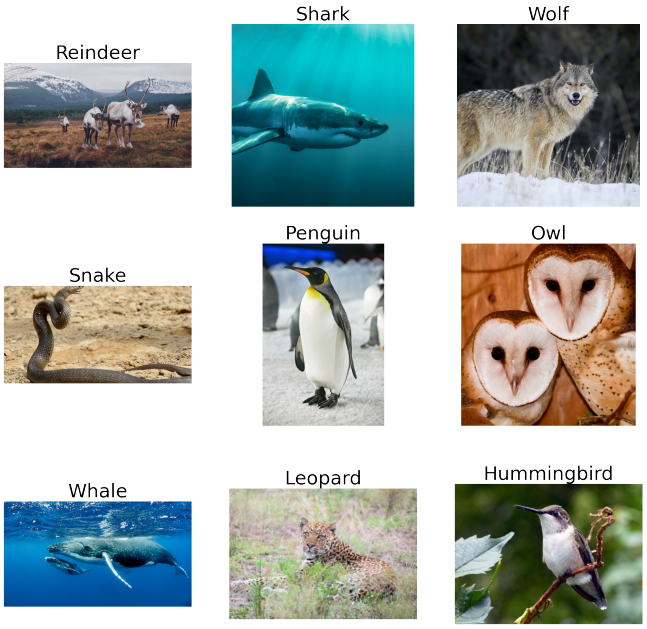

Figure 4. Colour Images Examples from 9 Classes (Banerje)

produced a gray-scale image collection based on the colour image dataset. Figure 5 illustrates nine classes from a total of ninety classes. Both the colour and gray-scale image datasets will be partitioned into train, validation, and test sets at a ratio of 0.6:0.3:0.1. The dataset should be quite difficult in that the amount of data points for each class to be trained should be fairly small. By creating two datasets, one with colour and one without colour, We can evaluate the performance of all updated models in a variety of scenarios in which the model is required to learn both colour and shape features or only shape features.

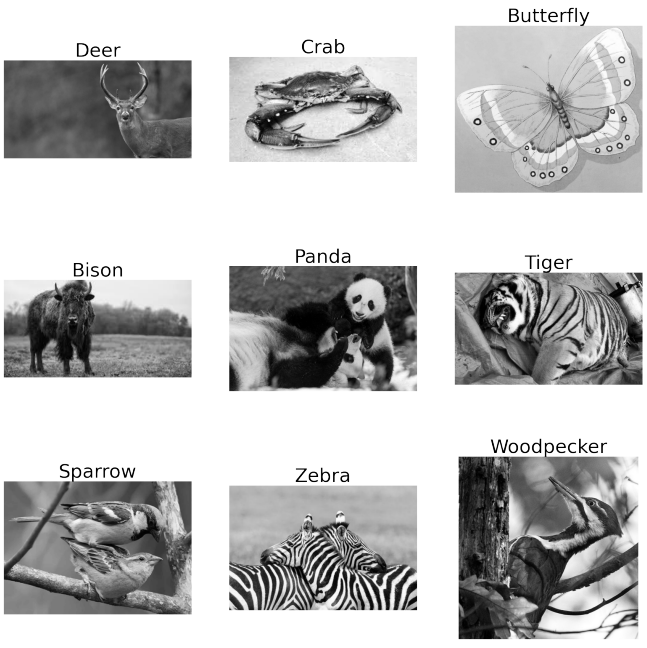

Figure 5. Grey Images Examples from 9 Classes (Banerje)

Outline: The following sections will describe how we con-

struct filters and modify AlexNet at first. Then we will show all of the outcomes of experiments, along with our analysis of them. Finally, we will reach a judgement regarding whether the results support or refute the hypothesis.

2. Methodology

This section will concentrate on the creation of filters and the modification of models.

2.1. Vision Methodology

The core structural elements in vision part are three different filters: Gabor Filters, Edge Filters and Blob Filters, respectively. They are inspired in part by Marr's Primal Sketch features (Marr, 1982) and in part by the features observed in the first layer of AlexNet (Krizhevsky et al., 2012). Three different kinds of filters all originate from some certain kernels. To generate these kernels directly we will rely on an external Python package named scikit-image (scikit image). In this section, more details about the four filters will be given. The filters will be used to replace the first layer of AlexNet.

2.1.1. Gabor Filters

The kernel of Gabor Filters is a Gaussian kernel modulated by a complex harmonic function. Harmonic functions consist of an imaginary sine function and a real cosine function (Fogel & Sagi, 2004). Their spatial frequency is inversely proportional to the harmonic wavelength and the standard deviation of a Gaussian kernel. The bandwidth is also inversely proportional to the standard deviation. In this paper we only adopt the real part of the Gabor kernel. The function we use to generate gabor kernels is **skimage.filters.gabor_kernel( )**. There are two arguments for variables: frequency and theta. We can construct a Gabor Filter of exact size 11x11 using a particular combination of frequency and theta. An example of Gabor filter with frequency being 0.27 and theta being $\pi/4$ is illustrated in Figure 6.

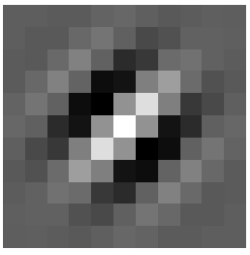

Figure 6. Gabor Filter in grey-scale with frequency = 0.27 and theta = $\pi/4$

Besides the standard Gabor Filter, we also constructed some adaptations of the Gabor Filters named Gabor Half Bar Kernels. For Gabor Half Bar Kernels, the same gabor kernels are applied and then the values of the bottom part of the kernels are flipped. Since the height and width of the gabor filters are both odd with the same value, the middle row of the filters is zeroed out to the filters symmetric. Figure 7 demonstrates the Gabor Half Bar Filter with frequency equal 0.10 and theta equal $\pi/4$.

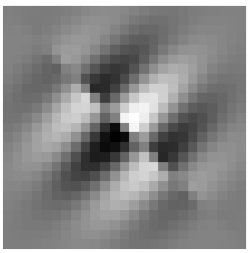

Figure 7. Gabor Half Bar Filter in grey-scale with frequency = 0.10 and theta = $\pi/4$

2.1.2. Edge Filters

Edge Filters, as the name suggests, are used to detect the edges in a figure. Gaussian kernels are used to construct edge filters and are generated from **skimage.filters.gaussian( )**. Two Gaussian distributions, one with the maximum and the other with the minimum, are applied to give strong response when the color changes from one to another. As there is no argument to rotate the Gaussian manually. For instance, if theta is 45 degrees, the centre of the Gaussian should be in the top-right position, thus:

0 0 1
0 0 0
0 0 0

In order to get a complete 2-dimensional Gaussian distribution, we need to find a proper matrix size with the sum of the matrix roughly equal to 1.0. When the sum condition is satisfied, it is better to ensure that the filters are normalised to 1.0. We need to set up the Gaussian filters with 'minimum' as well. If theta is 45 degrees, the centre of the Gaussian should be in the bottom-left position, thus:

0 0 0
0 0 0
-1 0 0

When we add the two corresponding filters together, we ought to get a filter with centres located like:

0 0 1
0 0 0
-1 0 0

After applying the Gaussian kernels to the centres, the ~~figure~~ should resemble Figure 8.

2.1.3. Blob Filters

We also use Gaussian distributions to recognise dots and circles in images when developing Blob Filters. As there are no alternative orientations for a dot or circle, the only variable is sigma. Both Gaussian centres are located in the filter's centre point, the first positive and the second negative. The sigma of the second Gaussian 1.6 times the

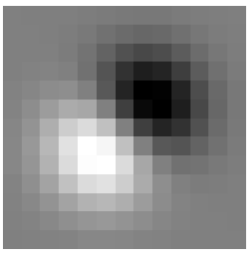

Figure 8. Edge Filter in grey-scale with sigma = 2.0 and theta = $\pi/4$

corresponding sigma of the first Gaussian. Take grey-scale Blob Filter with positive filter sigma being 2 and negative filter sigma being 3.2 as an example, the plot of the Blob filter is shown in Figure 9.

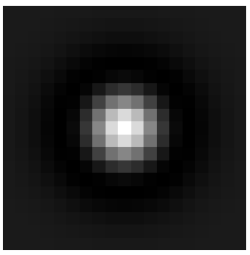

Figure 9. Blob Filter in grey-scale with sigma = 2.0 and sigma = $\pi/4$

2.2. Model Explanation

Flowing section will explain and shows What is the model we will use for examining our hypothesis.

2.2.1. Baseline Model

As a baseline model for the experiments, we prepared two Alexnet model. First one is Alexnet without any pre-training, and another one is Alexnet with pre-trained parameters downloaded from PyTorch (Pytorch_Team). ~~The architecture of the both baseline Alexnet model is same.~~ Only difference between these models are number of the middle units (number of ~~the~~ kernels) of the first layer. Due to the number of the kernels we could prepare~~d~~ (Gabor, Edge, Blob filters, etc.), the number of the kernels for the model without pre-trained are 48, while pre-trained one has 64 kernels. However, since other architecture are all same, the performances of these two model will not have big difference from architecture point of view. Also, since these models are only used for baseline, high performance is not expected. In the following part of the paper, we will call Alexnet without pre-train as **OriginalAlexnet**, and model with pre-trained as **PretrainedAlexnet**.

2.2.2. Compared Model

As the improved model which included our suggested filter for first layer kernels, we proposed four different model which will be able to examine the effect of our self defined filters. ~~These models are following.~~

- **SubAlexNet**:
  This model have exactly same architecture and number of parameters as the OriginalAlexnet. However, 48 kernel of the first layer is **substituted by our self defined kernels(Gabor, Edge, and Blob filters) before training**. After the substitution, the model will be trained on training set and validate on validation set. This model is defined on the assumption, "kernel in the first layer of the CNN will be similar to Gabor, Edge, and Blob filters", and expected to see faster convergence (stop of validation accuracy) of the training process, since the kernel for the first layer is already given.

- **AlexSubNet**:
  This model have exactly same architecture and numbers of parameters as the OriginalAlexnet. However, first 48 kernels will be **substituted to our self defined filters after the training**. After the substitution, the model will be evaluated on the test set and recorded its accuracy. According to our hypothesis, the kernels of the first layer after the training process should be similar to our self defined kernels, therefore, substituting these trained kernels with our generated kernel should not degrade the performance of the model so much. Since the model's training process are exactly same as OriginalAlexnet, there will be no any special difference in training performance (like speed of convergence) compared to OriginalAlexnet.

- **PreSubAlexNet**:
  This model have exactly architecture and number of parameters as PretrainedAlexnet. However first 64 kernels will be **substituted by our self defined kernels before the training**. After the substitution of the kernels, the model will be trained on training set and evaluated on validation set. By observing the kernels of the first layer of the pre-trained Alexnet, it is found out the content of the kernels are very similar to what we defined as our self defined kernels (see Figure 2). Therefore, we considered that substitution of these kernels to our self defined kernels will not have big impact on the performance of the model. Same reason with the SubAlexNet, since the kernels are already give before the training, the training process should be faster in convergence compared to OriginalAlexnet. In addition, since originally it was pre-trained model, not only first layers kernels are given but also another parameters are already given, its training convergence should be faster than SubAlexNet too. Since the PretrainedAlexnet is used as the base of this model, the accuracy performance are expected to be almost same as the performance of PretrainedAlexnet.

- **PreAlexSubNet**:
  This model will have exactly same architecture and number of parameters as PretrainedAlexnet. However, 64 kernels of the first layer will be **substituted by**

our self defined kernels after the training. After the substitution, the model will be evaluated on the test set and recorded its accuracy. From the observation stated in above, it is reasonable to say that the kernel's content of the pre-trained model is similar to our self defined kernels at least in the shape pattern. Therefore, we assumed replacements of the 64 kernels of the first layer of the trained model will not effect the performance of the model too much. The accuracy of the model on test set are expected to be almost same with the performance of the PretrainedAlexnet. Since, the substitution are done after the training process, the speed of training process will not have difference with training process of the PretrainedAlexnet which are base model of this model.

~~In the kernels substitution process, we used small technique to adjust the kernel size to required shape.~~ Since our self defined kernels are generated as shape (1,N,N), and kernels in the first layer of the model need to be the shape (3,N,N), we will required to increase the dimension of the generated kernels. To solve this problem, we defined a function which takes (1,N,N) shaped kernels as input, copy the kernels to three layers with random flip (inverting) of the kernel value, and stack these kernels as one kernel to form (3,N,N) kernel (see Figure 10). The reason for the random flipping of the kernels are from the observation of the first layer kernels of the pre-trained Alexnet. We found out the (3,N,N) kernels for the pre-trained Alexnet is constructed from one pattern kernel and its randomly inverted kernels (see Figure 3). By following this observed feature of the pre-trained model, we conducted the random inverting in the construction of the kernels to be substituted.

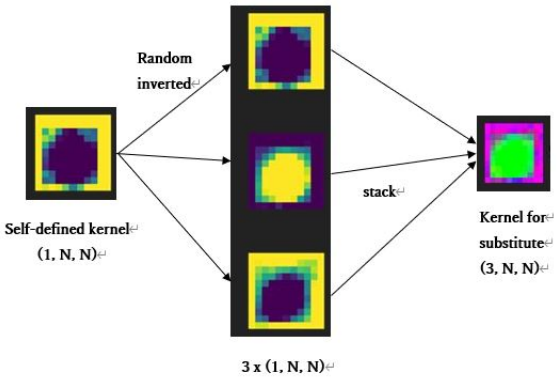

Figure 10. Process of converting self defined kernels for substitution

2.3. Experiment

In this section, we will explain 3 experiments we did for examining our hypothesis stated in introduction chapter. Training process are same for all the models. We used following high parameters for the train g process of ~~the~~ all models.

- Epoch number = 300
- Loss function = Cross entropy loss
- Optimizer = Adam
- Learning Rate = 0.0002

2.3.1. Experiment1

The purpose of this experiment are to evaluate the performance of our suggested models (SubAlexNet, etc.) compared with the baseline models. We will focus on the training process (number of epoch needed for convergence of the model) and performance of classification (accuracy value). First the baseline models, **OriginalAlexnet** and **PretrainedAlexnet** are trained and evaluated on training set and validation set respectively. The training loss, training accuracy, validation loss, and validation accuracy are logged for each epoch so that we will be able to plot it and find out the number of epoch needed for the convergence and whether the model over-fitted or not. After the training, as the performance of the model, the accuracy on the test set is computed. Next, the trained models OriginalAlexnet and PretrainedAlexnet from previous process will be changed into **AlexSubNet** and **PreAlexSubNet**. The kernels of the first layer of these models will be substituted with our self defined kernels and the test set accuracy is computed with these new parameters. The initial kernels of the first layer for **AlexSubNet** and **PreAlexSubNet** are shown in Figure 11 and Figure 12 respectively.

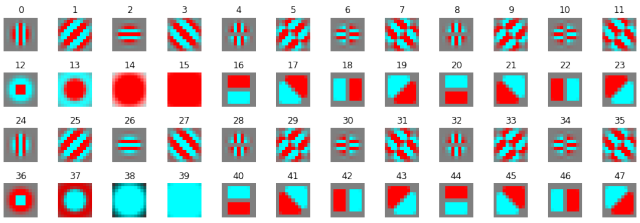

Figure 11. Initial kernels of first layer substituted by Gabor, Edge and Blob Filters in the **SubAlexNet** model

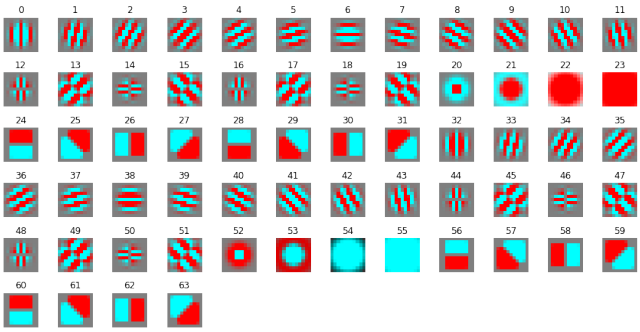

Figure 12. Initial kernels of first layer substituted by Gabor, Edge and Blob Filters in the **PreSubAlexNet** model

The computed test accuracy will be compared with accuracy of the original models. Lastly, we prepare **SubAlexNet** and **PreSubAlexNet** explained above. We will train these

models with same methods as OriginalAlexnet and PretrainedAlexnet training process. Again, training loss, training accuracy, validation loss, and validation accuracy are logged for each epoch for the purpose of detecting overfitting and number of epoch for the model convergence. After the training process, the performance of these models will be evaluated on test set and accuracy will be computed. The training process and test accuracy will be compared with corresponding baseline models.

2.3.2. Experiment2

The purpose of this experiments is to examine the performance of our self defined filters on classification task of the AlexNet on grey-scaled images. Since our self defined filter are theoretically good at detecting part of the shape pattern (edges, etc) in the images, we considered that the benefit of our self defined filters will be more obvious if we removed color contexts from the images model will process. If we need to state the limitation of our filter, we would suggest the ignorance of our self defined filter about the color of the images. Since our self defined filters are generated as (1,N,N) shaped kernels, it does not include the context information about the color (for example, normal filter of first layer of Alexnet will have shape of (3,N,N) which 3 are representing RGB color dimension). We tried to tackle this problem by *random inverting and stack* methods, however, as it relies on the random inverting, it is clear that the methods are not enough for capturing color context. To remove the effect of the color of the images to the our kernels convolution performance, in this experiments, we removed all the color context from images by applying gray-scaling.The models and process for this experiment is exactly same as **Experiment1** except for using grey-scaled images for training, validation and testing. Same as Experiment1, **OriginalAlexnet** and**PretrainedAlexnet** are used as baseline models, **SubAlexNet**, **AlexSubNet**, **PreSubAlexNet**, and **PreAlexSubNet** are used form comparison with the baseline models. The training process and test accuracy will be examined for examining effect of our self defined kernels.

3. Results & Discussion

Following section shows the results and its discussion about each experiments.

3.1. Experiment1

The test accuracy of the models and change in validation accuracy in the training process of the models are presented in below table1 and figure13.

From the observation of the test accuracy, it is very clear that substitute the kernels after the training is very bad idea. In both model **AlexSubNet** and **PreAlexSubNet**, the classification performance are significantly decreased compared to each baseline model **OriginalAlexnet** and **PretrainedAlexnet** performance. It is very reasonable phenomena to be observed, since the parameters for the model

Table 1. Accuracy of the classification on test set

Model	Color Accuracy	Grey Accuracy
OriginalAlexnet	0.3486	0.3739
PretrainedAlexnet	0.4346	0.4720
SubAlexNet	0.3692	0.3660
AlexSubNet	0.0111	0.0174
PreSubAlexNet	0.3602	0.3887
PreAlexSubNet	0.0063	0.0111

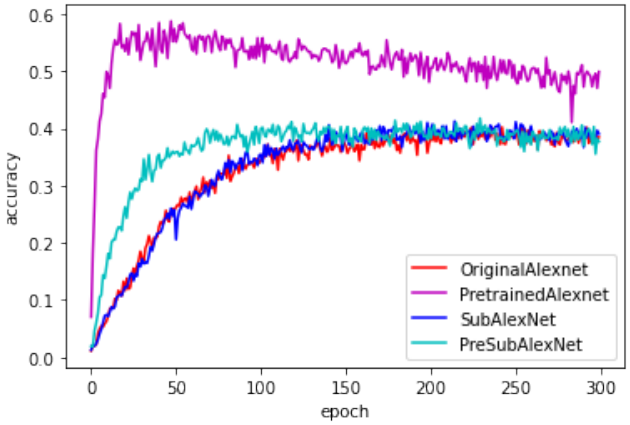

Figure 13. Each model's validation accuracy in training process

are carefully tuned during training process, therefore even replacing first layer's parameters will break the harmony built by whole the model parameters. From the training process plot, we find out actually the convergence speed of the pre-trained model is the best. It converges around 50 epoch, while second fast model **PreSubAlexNet** converges around 100 epoch, and **OriginalAlexnet** and **SubAlexNet** ~~are~~ converge~~s~~ around 150 epoch. By comparing the model **PreSubAlexNet** and baseline model **PretrainedAlexnet**, it became clear that introducing ~~of~~ our self defined kernels leads to deteriorated performance in both speed of convergence and accuracy. This is because the ability of the baseline **PretrainedAlexnet** is very high, and it is designed and trained for the transfer learning. Since our self defined kernels initialized are not capturing color context, it is difficult to beat the pre-trained model which kernels are trained to capture the color contexts. By comparing another pair of model OriginalAlexnet and **SubAlexNet**, we found the fact both performance on the speed of convergence and accuracy are almost same, although **SubAlexNet** has slightly higher accuracy than **OriginalAlexnet**. Since our assumption is "the kernels in first layer of the model will be similar to our self defined kernels", the small change in the accuracy performance is not surprising. However, it is surprising to see that speed of convergence is same, since we expected faster convergence due to pre-given first layer kernels which ease should ease the parameter learning process. As a possible reason, we suggested our kernel's ignorance about the color context. Since the initialized kernel are randomly initialized its color aspect, the model took almost same time to learn these missing color contexts of the inputted images.

3.1.1. Filter

To have a better understanding of how the first layer filters will evolve during training, we extract the first layer kernels and plot them in a colour image as follows. The kernels of first layer in the **OriginalAlexNet** model after trained for 300 epochs are shown in Figure 14. As seen from Figure 14, all the kernels show pretty random colours. As a result of the OriginalAlexNet model's outcomes, we declare the assumption that the forms of the kernels in AlexNet's first layer all resemble some pre-defined filters is wrong. It appears that the only way to generate regular filters is to train the AlexNet on a very large and thorough image dataset.

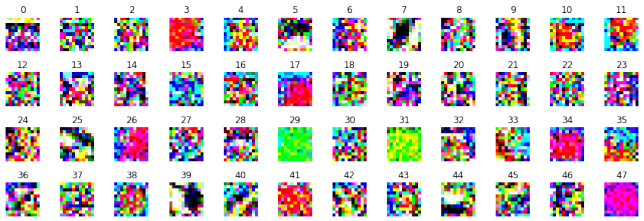

Figure 14. kernels of first layer in the **OriginalAlexNet** model after trained for 300 epochs

The kernels of first layer in the **PretrainedAlexNet** model after trained for 300 epochs are shown in Figure 15. The shapes and colours of the kernels have remained rather constant, which correlates to the findings in Figure 13 showing the **PretrainedAlexNet** model learns at a breakneck pace.

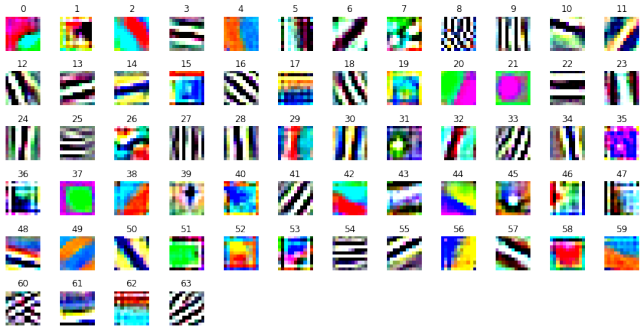

Figure 15. kernels of first layer in the **PretrainedAlexNet** model after trained for 300 epochs

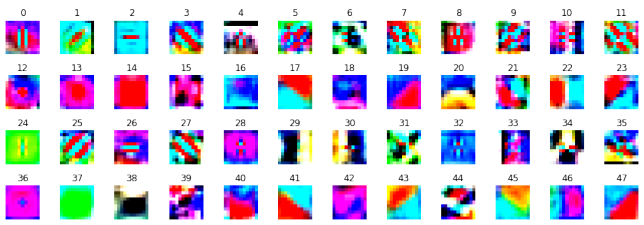

Figure 16. kernels of first layer in the **SubAlexNet** model after trained for 300 epochs

After trained for 300 epochs, the kernels of first layers in the **SubAlexNet** model and the **SubAlexNet** model and the **PreSubAlexNet** model are shown in Figure 16 and Figure 17. Compare with the initial kernels that are illustrated in Figure 11 and Figure 12, we can see that some kernels retain their original forms roughly. Though both models, **OriginalAlexNet** and **SubAlexNet**, have the same overall learning curve as seen in Figure 13, their first layer kernels are substantially different in shape.

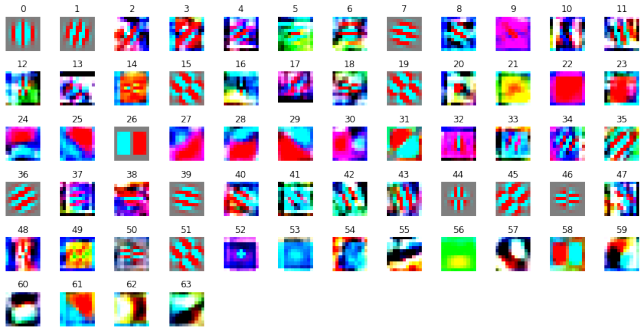

Figure 17. kernels of first layer in the **PreSubAlexNet** model after trained for 300 epochs

3.2. Experiment2

The test accuracy of the models and change in validation accuracy in the training process of the models are presented in below table1 and figure18.

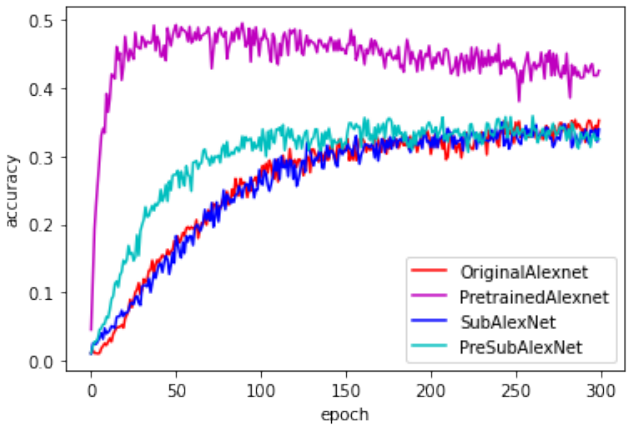

Figure 18. Each model's validation accuracy in Grey-scaled training process

The performance results on speed of convergence and accuracy are almost same with results of the Experiment1. Again, the low accuracy of the **AlexSubNet** and **PreAlexSubNet** tells us that the replacement of the kernels after the training is poor methodology. Different part from the Experiment1 is, **the speed of convergence of the PreSubAlexNet become slower and became closer to performance of OriginalAlexnet and SubAlexNet**. This shows that the **PreSubAlexNet** could not make effective use of the pre-trained parameter after the layer2 which are trained under the existence of color. However, since the performance of the **PretrainedAlexnet** is still high and the performance of the **OriginalAlexnet** and **SubAlexNet** are almost same,

we could not observe any benefit of introducing our self defined kernels. It is very unexpected to see almost same performance relationship as colored experiment in this experiment which uses grey-scaled images to avoid model to use color context of the inputs. In our expectation, since our self defined model should be better in detecting part of shape instead of color, by removing the color components from input, the model with our self defined kernels should suppress the baseline model which requires color contexts as one of the input features. Since the experiment are conducted under the grey-scaled data, we could not make color context as excuse to this results. We should carefully observe the visualized kernels and determine whether the kernels of the first layer of the model can be substituted by our self defined kernels.

3.2.1. Filter

As we did in Section 3.1.1, we also showed all of the first layer kernels for each model. The results are mostly consistent with what we saw in the colour scenarios.

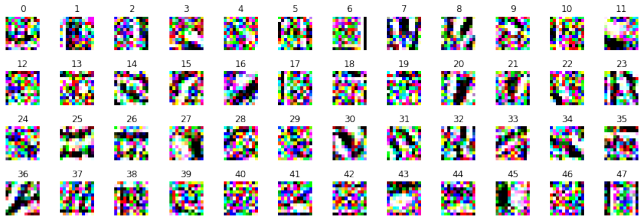

Figure 19. kernels of first layer in the **OriginalAlexNet** model after trained for 300 epochs

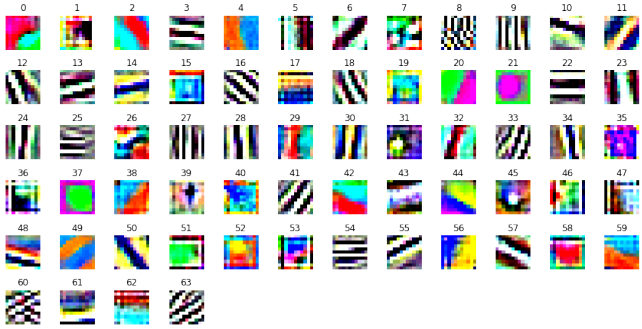

Figure 20. kernels of first layer in the **PretrainedAlexNet** model after trained for 300 epochs

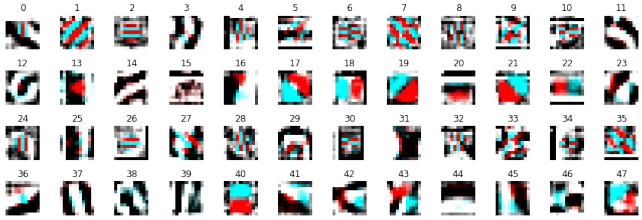

Figure 21. kernels of first layer in the **SubAlexNet** model after trained for 300 epochs

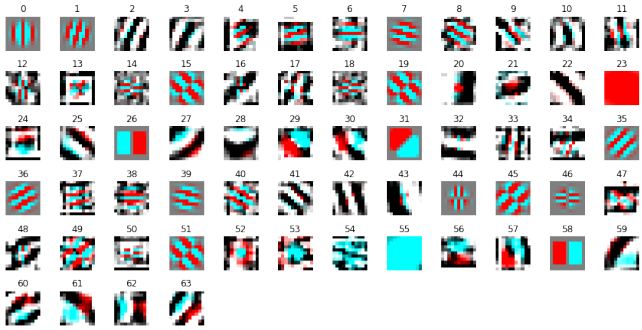

Figure 22. kernels of first layer in the **PreSubAlexNet** model after trained for 300 epochs

All of the preceding results demonstrate that changing images from colour to grey-scale has no effect on either the convergence epochs or the accuracy rate.

4. Conclusions

Based on our findings, we would conclude that modifying the untrained kernels of AlexNet has no effect on training speed or accuracy. Indeed, we observed no discernible differences. A possible reason is that because modification on the only first layer has little or no effect on the optimiser, these pre-defined kernels are of no benefit to the optimiser. The only difference could be if the first layer maintains a distinct local optima, but this is also ineffective. Meanwhile, altering the first layer of a pre-trained AlexNet may have a deleterious consequence. This demonstrates that a huge dataset is still far more useful for deep net training. Setting some pre-defined kernels is not a quick way to accomplish this.

References

Structural health monitoring of civil infrastructure systems / edited by Vistasp M. Karbhari and Farhad Ansari. Woodhead Publishing in materials. Woodhead Pub., Cambridge, UK, 2009. ISBN 9781615831098.

Aach, Til, Kaup, André, and Mester, Rudolf. On texture analysis: Local energy transforms versus quadrature filters. *Signal Processing*, 45(2):173–181, 1995. ISSN 0165-1684. doi: https://doi.org/10.1016/0165-1684(95)00049-J. URL https://www.sciencedirect.com/science/article/pii/016516849500049J.

Banerje, Sourav. Animal image dataset. https://https://www.kaggle.com/datasets/iamsouravbanerjee/animal-image-dataset-90-different-animals. Accessed: 2022-03-29.

Fogel, I. Y. and Sagi, Dov. Gabor filters as texture discriminator. *Biological Cybernetics*, 61:103–113, 2004.

Goodfellow, Ian. *Deep learning / Ian Goodfellow, Yoshua Bengio and Aaron Courville.* Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts, 2016. ISBN 0262035618.

Gustafsson, Fredrik. Particle filter theory and practice with positioning applications. *IEEE Aerospace and Electronic Systems Magazine*, 25(7):53–82, 2010. doi: 10.1109/MAES.2010.5546308.

Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In Pereira, F., Burges, C. J. C., Bottou, L., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Marr, David. *Vision*. W.H. Freeman and Company, 1982.

Pytorch_Team. Alexnet pytorch. https://pytorch.org/hub/pytorch_vision_alexnet/. Accessed: 2022-03-29.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. Imagenet large scale visual recognition challenge, 2014. URL https://arxiv.org/abs/1409.0575.

Sangaiah, Arun Kumar. *Deep Learning and Parallel Computing Environment for Bioengineering Systems*. Elsevier Science Technology, San Diego, 2019. ISBN 0128167181.

scikit image. scikit-image. https://scikit-image.org/. Accessed: 2022-03-29.

Snyder, Jeffrey A and Fedorovskaya, Elena A. Digital image processing and analysis: Human and computer vision applications with cviptools, second edition, by scott e. umbaugh. *Journal of electronic imaging*, 20(3): 039901–039901, 2011. ISSN 1017-9909.