Aimee Johnston
NEUR189B: Artificial and Biological Vision
Prof. Yuqing Zhu
Final Project Paper

**Introduction:**

For this final project, I wanted to investigate the generalizability of learned color features in a simple convolutional neural network (CNN). I was specifically interested in exploring whether or not a learned foreground feature (ie, a central object) could then be applied to the same object in the background of another image.

I decided to use color as a feature for a few reasons. Gowda and Yuan (2019) found that the color space of images does affect accuracy for image classification algorithms, indicating that colors are an important feature learned by CNNs. In addition, Cui et al. (2022) found that a CNN could be trained to accurately represent truck color. While this might seem like a simple issue, truck color is one of the more difficult color recognition problems, as trucks tend to not be in the same shape or positions as other vehicles, and are more affected by factors such as dirt. This shows that CNNs can effectively generalize color recognition, even in more complex environments, giving strong indication that a CNN would be able to generalize color from foreground to background.

There is also evidence supporting the use of red for this experiment. For biological vision systems, red tends to be a color that stands out, but that is also the case for CNNS. Hickey and Zhang (2020) found that, much like humans, CNNs exhibit hierarchical color processing, meaning that certain colors, specifically red, are learned before other colors, such as brown. Their research also demonstrated that CNNs perform better on image classification tasks if the image contains red compared to other colors.

I hypothesized that, if the learned convolutions were generalizable from foreground to background, then the network would show similar accuracy on images with red in the foreground compared to images with red in the background. In addition, I hypothesized that, if the features truly were generalizable, swapping the training and test data sets would show similar performance to the previous experiment.

**Methods:**

In order to test my hypotheses, I sourced images from the CIFAR-100 dataset (Krizhevsky 2009). The CIFAR-100 dataset is a labelled subset of the 80 Million Tiny Images Dataset (Korralba, et al. 2008), and contains 6000 32x32 pixel images with pre-assigned object labels. The Tiny Images Dataset has since been removed from public availability due to biased labelling, but the CIFAR-100 dataset is independently labelled with 100 object classes. For the purposes of this experiment, I did not use the pre-existing labels. Instead, I displayed the images and hand-sorted them into one of three categories based on the presence of red objects in the image.

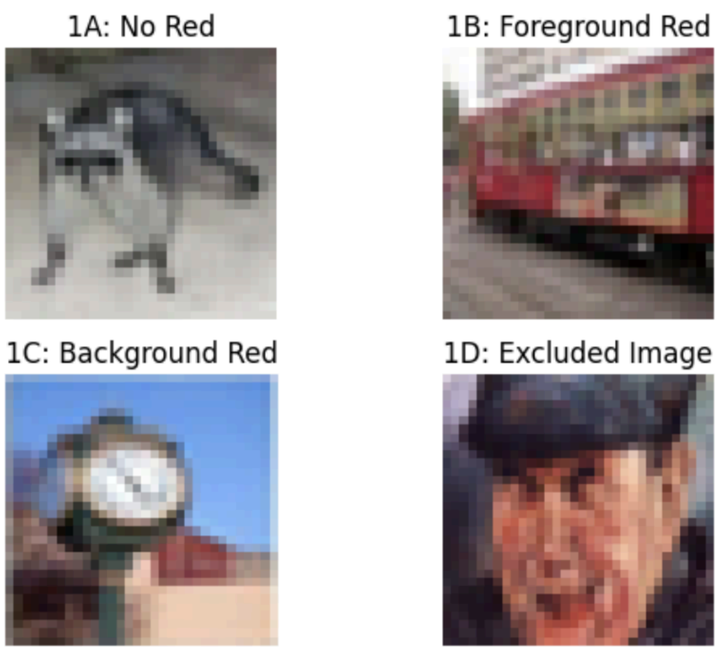



*Figure 1A-1C shows a sample image from each of the three image categories, while Figure 1D shows a sample image that was excluded from the dataset. The image in Figure 1D was excluded due to the presence of red patches on the man's face.*

These categories were: no red (see Figure 1A), red in the foreground (see Figure 1B), and red in the background (see Figure 1C). I also chose to exclude some images (see Figure 1D) from the dataset based on the presence of red without the presence of red objects, in order to avoid confusing the model.

To build my dataset from these images, I hand-labelled the first 850 images included in the CIFAR-100 training dataset. Of these 850, 61 had prominent/foreground red, 39 had background red objects, and 34 were excluded to avoid confounding data. I then created two datasets: a “train” dataset, containing the foreground red images and half of the images without red, and a “test” dataset, containing the background red images and the other half of the images without red.

While building my CNN (Red-Net), I wanted to keep the model as simple as possible, in order to prevent the network from learning more complex features. For this experiment, the goal was to test the generalizability of simple features, so I limited it to a fairly basic structure of one convolutional layer of 8x8 convolutions, one max pooling layer, a dense layer of two neurons, a ReLU layer, and finally a softmax layer.

```
Model: "sequential_72"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_71 (Conv2D)          (None, 25, 25, 4)         772

 max_pooling2d_69 (MaxPooli  (None, 12, 12, 4)         0
 ng2D)

 flatten_69 (Flatten)        (None, 576)               0

 dense_68 (Dense)            (None, 2)                 1154

 re_lu_65 (ReLU)             (None, 2)                 0

 softmax_68 (Softmax)        (None, 2)                 0

=================================================================
Total params: 1926 (7.52 KB)
Trainable params: 1926 (7.52 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

*Figure 2 shows a sample CNN architecture from this experiment, detailing the data output shape at each layer and the number of trained parameters. Figure 2 reflects the architecture for the network with 4 convolutions.*

Because of my small dataset, I was not satisfied with only one train and test accuracy metric, so to confirm my results, I ran the experiments eight times, each time using an increased number of convolutions. The first iteration used only one 8x8 convolutional filter, while the second used two, and so on up to eight. In order to compile and fit my models, I used the Adam optimizer that is built into the Keras package, and evaluated loss using binary cross-entropy.

**Results:**

Training and testing Red-Net as described in the methods section yielded somewhat unexpected results. As shown in Figure 3, training the model on foreground red showed a decent average train accuracy across network structures, but it consistently showed better performance on the test set, which contained the background red objects.

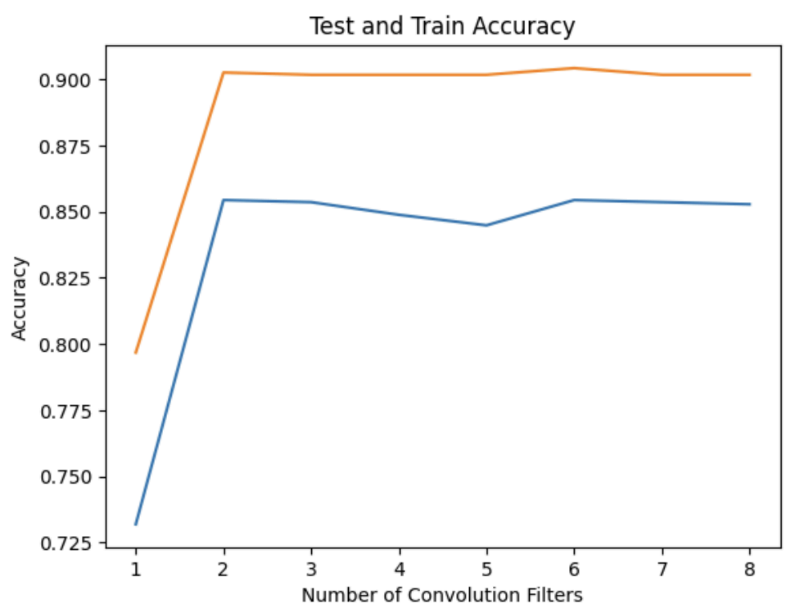
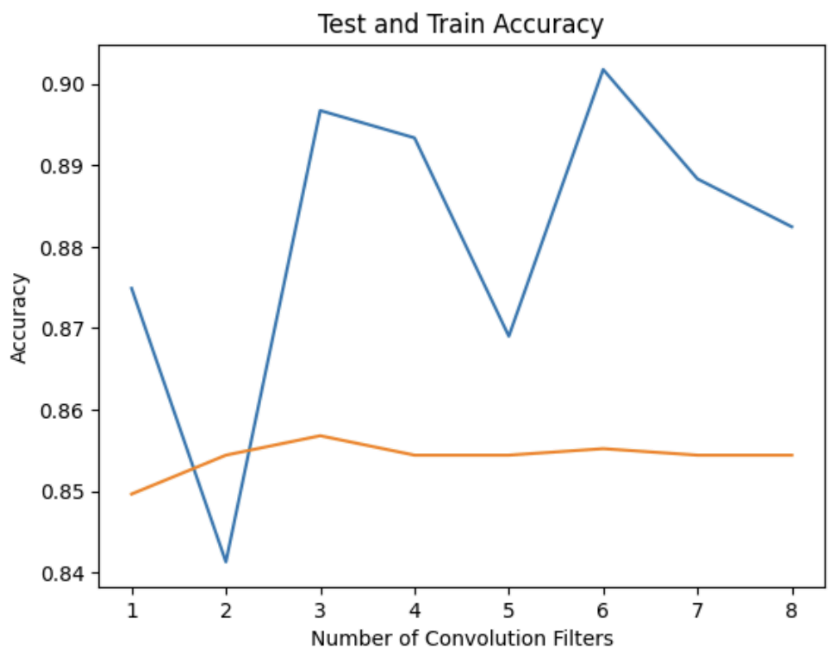



*Figure 3 shows the train (blue) and test (orange) accuracies for training on foreground red and testing on background red. For each network architecture, three epochs were used and the accuracies were averaged across epochs.*

This disparity in train and test accuracies indicates that the network is much more successful at detecting red on a smaller scale.

Swapping the train and test sets only further confirms this. As shown in Figure 4, when the model was trained on the background red images, the train accuracy was consistently better than the test accuracy. It was certainly more inconsistent than the previous experiment, but the general pattern still remained: the performance was better on the background red images than on the foreground red images.

*Figure 4 shows the averaged train (blue) and test (orange) accuracies for training on background red and testing on foreground red.*




**Discussion:**

Initially, these unexpected results were slightly perplexing, as train accuracy generally tends to be higher than test accuracy, except in the case of strong regularization. While both train and test performance were well above the 50% chance level, there was still a notable difference between the two in both experiments.

However, considering the potential convolutional features of the network can provide some insight into these results. Red-Net utilizes 8x8 convolutional filters, on an image that is 32x32 pixels. While the difference between the two sizes is not that significant, the convolutional filters are still smaller than the original image. Considering how coarse-grained the original images are, it would only take a few red pixels to even occupy the foreground of an image. It therefore makes sense that convolutional filters might learn an activation of just a few red pixels, which could work just as well on background objects. That being said, the results do seem to confirm my hypothesis: detecting red objects in the foreground is a task that can transfer to detecting the same color in the background.

While this experiment provided some insight into my hypothesis, it certainly had limitations, and there are a few extensions that could be used to confirm my results. The biggest limitation of this experiment is the dataset. Due to the time constraints of this project, I was limited in the amount of images I could label. Labelling was also a very subjective process, was completed over multiple sessions, and was not validated by any other individuals. Because of this, the category labels were likely somewhat inconsistent, and that could easily affect the training process with such a small dataset. I also did not do any padding of the images before convolution, and would be interested to see how that might affect the results.

In order to make any broader statements about Red-Net's color detection, it would also be necessary to look at the specific convolutional filters learned, both when trained on red in the foreground and when trained on red in the background. Examining these features is key to understanding if Red-Net is operating as intended, or learning some other unknown pattern that can yield significant accuracy.

Another possible extension would involve spending more time manipulating the training input images, and only training the network on grayscale and red images (red objects with every non-red part in grayscale), and then testing it on fully colored images. This would help further ensure that the network would only learn red-specific color features. A transfer learning approach would then be used in order to generalize the network's color recognition process.

However, this approach could cause the network to learn several filters that are just an all-red activation, which could still be activated by any pixels with a substantial red color channel value.

If these extensions confirmed my experimental results, there would be exciting implications for CNN research. One related paper, Tanmayi et al. (2023), used a pre-trained neural network (AlexNet) to detect fires in static images, and specifically mentioned the red color component as essential to first detection. If a network could be trained on images with forest fires in the foreground, and generalize that to detecting smaller fires in larger images, that would be an incredible environmental resource.

**References:**

Cui, J., Kang, C., Ma, X., Cao, R., Chen, X., Li, H., Zhang, J., Zhao, C., Ma, T., & Wang, X. (2022). Research on Recognition Method of Truck Colors Based on Convolutional Neural Network Model. In *Proceedings of the 14th International Conference on Computer Research and Development (ICCRD)*. IEEE. DOI: 10.1109/ICCRD54409.2022.9730465.

Gowda, S. N., & Yuan, C. (2019). ColorNet: Investigating the importance of color spaces for image classification. In *Computer Vision–ACCV 2018: 14th Asian Conference on Computer Vision, Perth, Australia, December 2–6, 2018, Revised Selected Papers, Part IV* (pp. 581-596). Springer International Publishing.

Hickey, C., & Zhang, B.-T. (2020). Hierarchical Color Learning in Convolutional Neural Networks. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (pp. 1559-1562). IEEE. DOI: 10.1109/CVPRW50498.2020.00200.

Krizhevsky, A. (2009). Learning Multiple Layers of Features from Tiny Images. Technical Report, Department of Computer Science, University of Toronto.

D. Tanmayi, E. Udaya and P. B. Pati, "Detecting Fire in Color Images using Convolutional Neural Network Architectures," *2023 IEEE 8th International Conference for Convergence in Technology (I2CT)*, Lonavla, India, 2023, pp. 1-6, doi: 10.1109/I2CT57861.2023.10126490.

Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: a large dataset for non-parametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 30(11), 1958-1970.