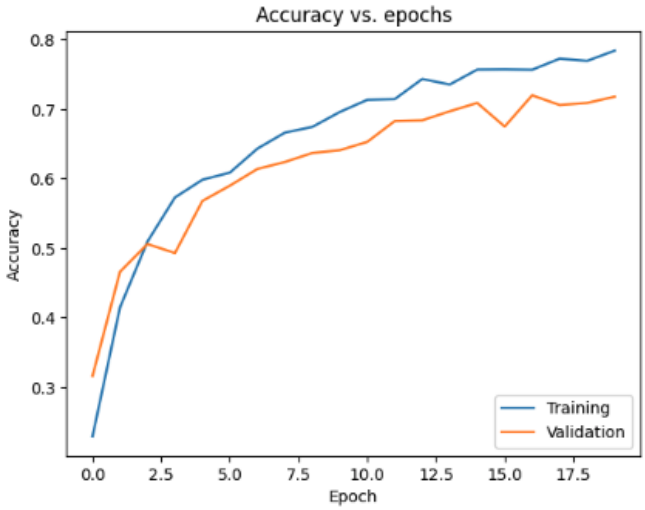
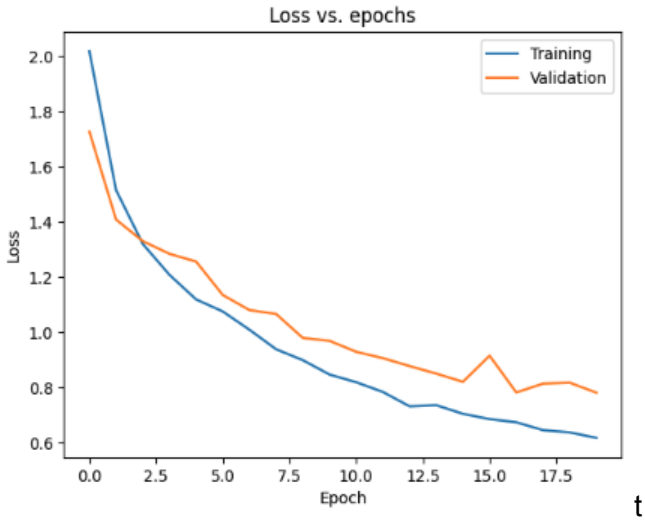
Ben Harris

Part 1:

Part 1 (Application to the EuroSat dataset): Base Model (comprising of 6 layers)
Training History Plots

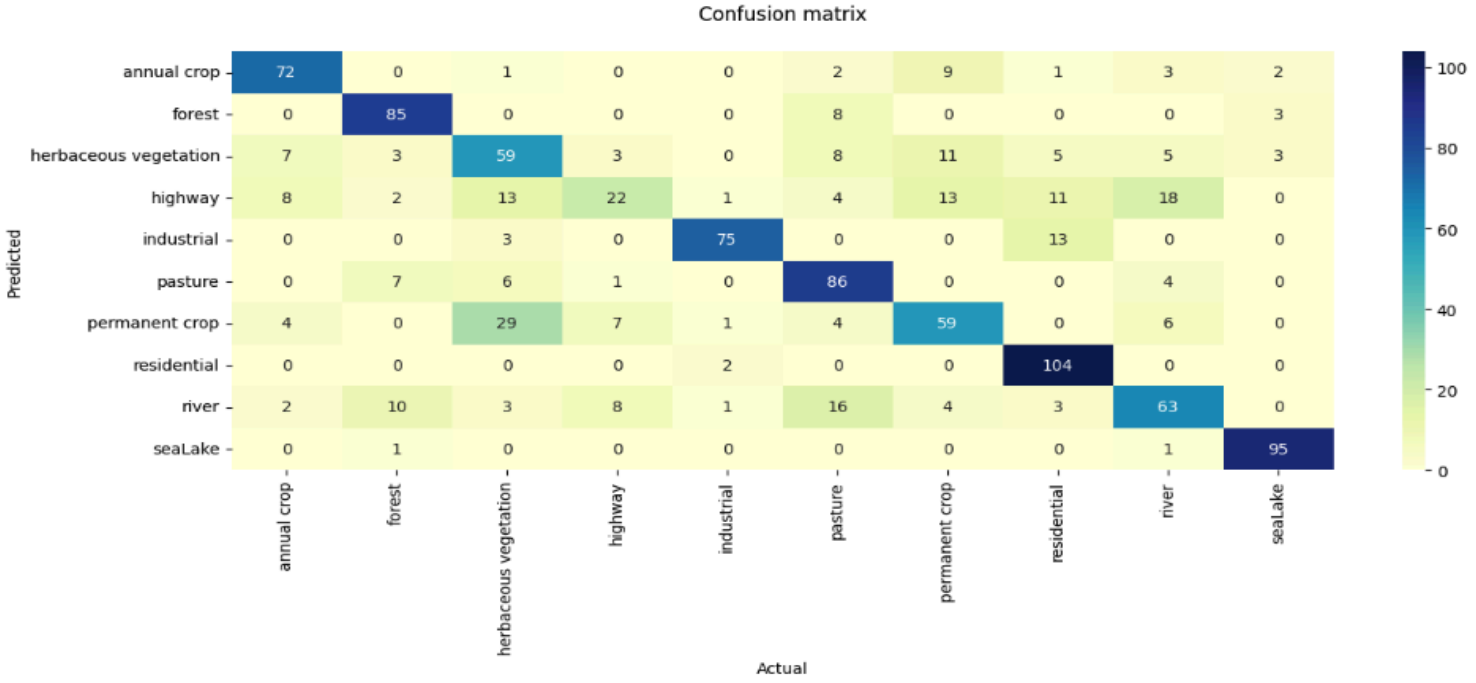




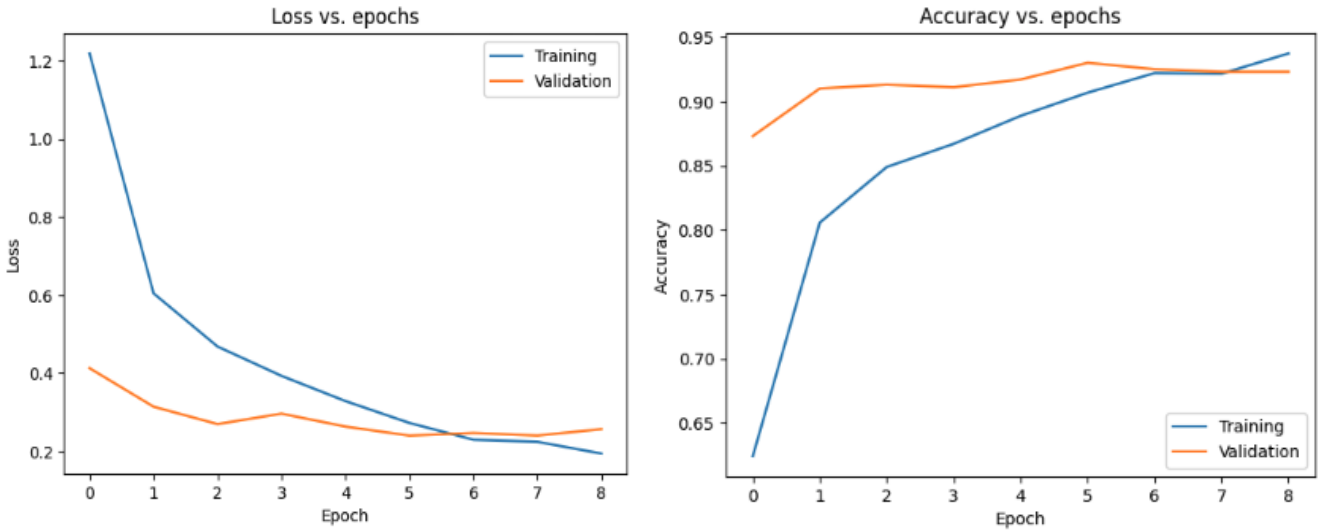
t

Final training accuracy value: 0.7843
Final test accuracy value: 0.7180

Part 1 (Application to the EuroSat dataset): Base Model (comprising of 6 layers)



Part 1 (Application to the EuroSat dataset): Transfer Learning Based Model
Training History Plots



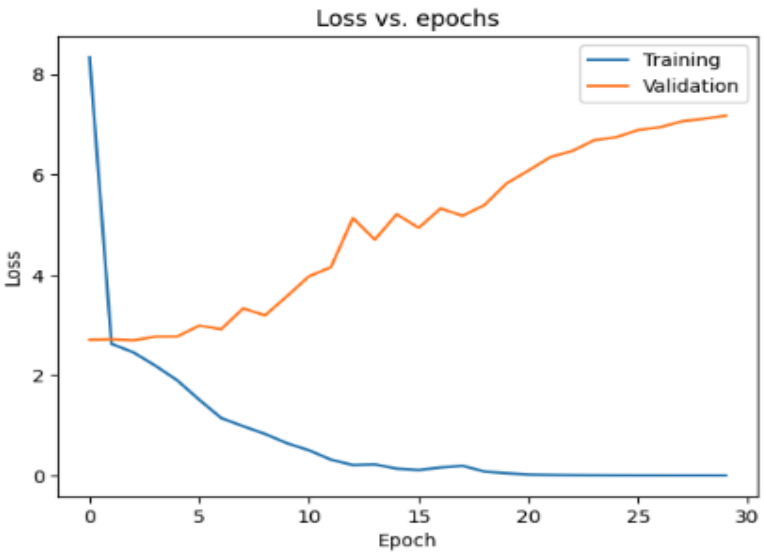
Final training accuracy value: 0.9373
Final test accuracy value: 0.9230

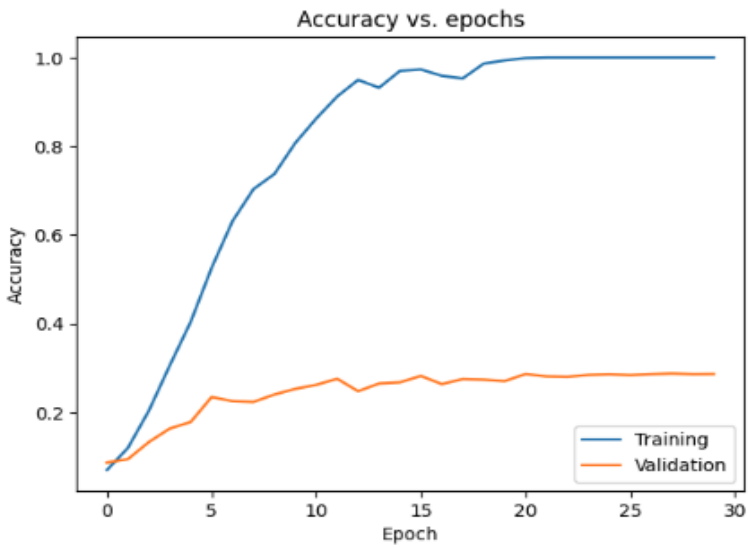
Part 1 (Application to the EuroSat dataset): IMPROVEMENTS TO THE MODEL
Describes the idea I incorporated to improve the performance and indicates the changes and parameter values. Also includes the loss, accuracy, and confusion matrix.

I added one more dense layer to my model with only 32 nodes and made my dropout only 0.2 along with a kernel regularizer of .001. I tested this model with a very small learning rate .00001 on 100 epochs. I started increasing my learning rate ever so slightly I noticed it was having trouble converging and the validation loss was not changing in several epochs so I added ReduceLROnPlateau to aid in better convergence for my loss function.

Part 2:

Part 2 (Application to the Scene dataset) : SimpleNET Results
Training History Plots





Final training accuracy value: 0.9640
Final test accuracy value: 0.2781

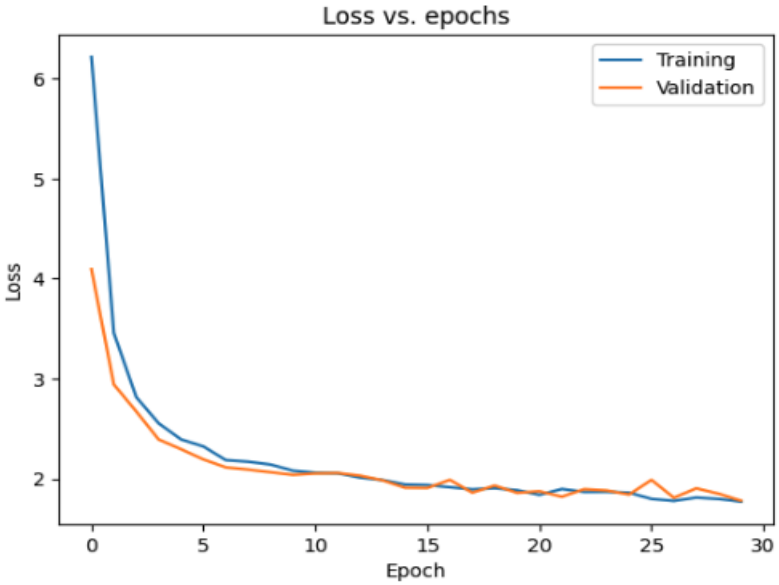
Part 2 (Application to the Scene dataset) : SimpleNET Results

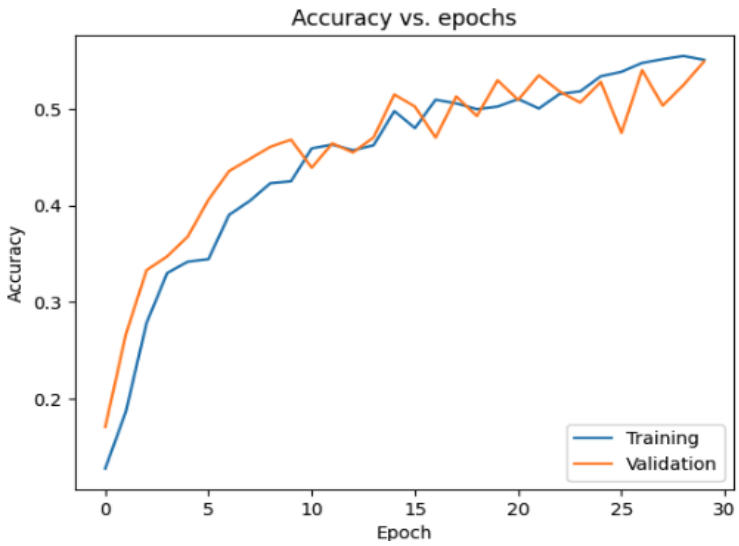
Experiment: Played around with some of the parameters in SimpleNET, and reported the effects for: 1. target_image_size, 2. kernel size of convolution filter; 3. Size of Maxpooling layer; Provide observations for training time and performance.

Target_image_size: Giving a bigger target image size without changing the existing model just exacerbated the previously overfitting model by allowing the model to extract even more features from the given image size. For an image size of (128,128) this resulted in a training accuracy of .999 and validation accuracy of 2.06. While it could also be seen that reducing the image size lessened this effect dramatically (32,32) resulted in a training accuracy of 0.2567 and validation accuracy of 0.1966 not allowing the convolutional layers to extract enough features from the given input size and showing that it would require a more complex model to represent it, given this input size. It should be noted a larger input size would make my fitting my model take longer due to it needing to extract more features.

Conv2D Kernel size: If a larger kernel size is used such as (7,7) it requires either the padding=”same” to maintain the dimensions of the feature maps after convolution or adjust the pool size to a (2,2) to accommodate. When doing this it would result in more computational usage but also the extraction of larger scaled features in the input data because it is now allowed to find patterns in larger regions. The results of the above kernel size being used with a training accuracy of 0.8861 and validation accuracy of 0.2074. With a kernel size used such as (3,3) and pooling of (3,3) we see overfitting once again but also an increase in the validation accuracy. This is seen in a training accuracy of 0.9817 and validation accuracy of 0.2817. Just as above in changing to a larger input size, here having a larger kernel size wouldn't be too noticeable but would make fitting my model take longer due to it needing to extract features from a larger size.

Part 2 (Application to the Scene dataset) : REGULARIZED SimpleNET
Training History Plots





Final training accuracy value: 0.5946
Final test accuracy value: 0.5860

Part 2 (Application to the Scene dataset) Reflection of Regularized mode
What regularization techniques I used and specified where I inserted dropout or batch normalization layer; value of drop out rate, etc.)

I first augmented my data and then used batchNormalization before my Convolutional layers. Then I used a .01 L2 regularization on both convolutional layers as well as the dense layers to help maintain smaller weights distribution being dealt to the neurons in the layers and keep consistent weight distribution. I placed dropout layers of .2 after both convolutional layers as well as the dense layers as this is where I saw the most benefit.

Compared the loss and accuracy for training and testing set, between regularized and un-regularized model and interpreted this result.

The regularized model gave more generalized conclusions because it was better for unseen data (validation data), as well as granting a more stable validation loss and accuracy. In contrast, the unregularized model is too focused on the training data, and this leads to poor generalization as seen by its validation performance. Regularization techniques thus help to achieve a model that performs well better on the validation data by not overfitting to the testing data by maintaining weights in a more consistent pattern. Dropout also helped to not overfit the training data.

Part 2 (Application to the Scene dataset): Transfer Learning Based Model
What the components of my transfer-learning based model were and how many dense layers after the pre-trained model. How many layers were frozen in the pre-trained model. Etc.)
Reflection: what does fine-tuning a network mean?
Reflection: why do we want to “freeze” some layers of the pretrained network? Why can we do this?

I added two dense layers after the pretrained model, one was with 64 features and the other for the output layer of 15. I froze all but the last 80 layers of the pretrained model.
Fine tuning the model was a process of first taking an existing model, or one you have developed yourself, that does an adequate job at fitting to the training data (helpful if slightly underfitting). Then manipulating the factors in the layers to help the model fit the data and generalize from it better. This often becomes a back and forth stride from changing a parameter slightly and noticing changes when fitting to the dataset. This can be done through regularizers, learning rate, dropout rate, and more. When it comes to fine tuning it was important for me not to change too much of anything too dramatically, (like adding or removing whole layers) in between testing the model and how it fit the dataset.
When it came to freezing layers It became very useful to freeze certain amounts in order to utilize the learned weights that capture many of the general features necessary in our new model. It aslo became very useful to unfreeze layers at the end of the pretrained network because these are were the more specific features are captured depending on the use-case of the model. For own use we could unfreeze and train the last layers of the pre-existing model to capture more features specific to our existing dataset. We can do this because early layers focus on features that are very general such as edges and shapes and can be utilized for basically all images.

Conclusion: Briefly discusses what I have learned from this project.

This project provided valuable insights into building and fine-tuning machine learning models, particularly in addressing overfitting challenges. I learned that working with an overfitted model made it harder to correlate validation accuracy improvements with parameter adjustments compared to an underfitted model. By adding layers, adjusting dropout rates, and using kernel regularizers, I significantly improved model performance on the EuroSat dataset. Techniques like ReduceLROnPlateau also proved essential in improving model convergence. In the Scene dataset, adjusting parameters such as image size, kernel size, and max-pooling helped balance feature extraction and computational efficiency.
Additionally, regularization techniques like batch normalization, L2 regularization, and dropout were key in improving model generalization and validation accuracy. Transfer learning taught me the value of freezing pre-trained layers to leverage general features while fine-tuning specific layers for task-specific improvements. This project enhanced my ability to analyze model performance through metrics, optimize models using advanced techniques, and develop models that generalize better on unseen data.