USING CNNS TO FIND FESTIVITY IN KNITWEAR

Table of Contents

Table of Figures

Acronym Dictionary

CNN:	Convolutional Neural Network
VGG :	Visual Geometry Group
DNN:	Deep Neural Network
ILSVRC:	ImageNet Large Scale Visual Recognition Challenge
ReLu:	A Rectified Linear Unit

Abstract

As our lives become increasingly digital, wardrobe digitisation and styling apps are becoming more popular. This report aims to train an image classification model to classify Christmas jumpers from everyday knitwear. Models explored include CNNs trained from scratch with a range of depths and pre-trained CNNs: AlexNet, VGG and GoogLeNet. To reduce overfitting all models were run on a greyscale version of the dataset, with success. The final model was a finetuned version of the GoogLeNet architecture achieving an accuracy of 93.67%. Exploring the misclassified examples exposed flaws in the dataset and suggest that when tested with a more robust dataset this accuracy can be expected to increase.

Introduction

As technology infiltrates more aspects of our lives, a rise in styling apps enables people to digitise their wardrobes and provides outfit recommendations.

Smart Closet, a styling app developed by Rabbit Tech Inc, currently has over 1 million downloads in both the apple app store and google play stores (Rabbit Tech Inc., 2023)with similar concepts being added to these stores regularly. This suggests a growing, increasingly competitive market.

This report aims to develop a Christmas jumpers classification model that gives a styling app a competitive advantage by ensuring outfits containing Christmas jumpers are only recommended within a certain date range.

Related Works

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) (Russakovsky et al., 2015) is an annual competition that sparked major developments in computer vision. This literature review focuses on the 2010-2014 classification tasks. The dataset includes tens of millions of high-resolution coloured images labelled with a thousand object categories (Deng et al., 2009).

In 2012, AlexNet was the first CNN to successfully use raw image data as inputs for a large-scale dataset. This removed the need for manually created features. AlexNet reduced the classification error from 26.2% to 16.4% and revolutionised the way CNNs were designed going forward (Krizhevsky et al., 2012).

The next major development was created by the Visual Geometry Group (VGG) at the University of Oxford. The VGG network is very deep, using many convolutional layers with small filters. This enabled it to identify more complex features and reduced the error rate to 7.3% (Simonyan and Zisserman, 2015).

Finally, Google developed GoogLeNet (Szegedy et al., 2015), using their inception model architecture to mitigate the vanishing/exploding gradients problem. GoogLeNet achieved the best error rate of the ILSVRC at 6.66% (Russakovsky et al., 2015b).

These architectures are further explored in Appendix A.

Another relevant baseline classification dataset is the Fashion-MNIST dataset (Xiao et al., 2017) which offers 70,000 greyscale images of clothing labelled by garment type (Trousers, Dress, Coat, etc.). Works relating to this dataset have been excluded from this literature review due to the low-resolution nature of the images.

Original Research Contribution

Initial literature reviews did not discover any attempts at thematic or seasonal clothing classification. Therefore, there is currently no standardised process for determining the festivity of clothing. Hence, the proceeding work is entirely original as it attempts to close this gap.

Dataset description

Dataset creation

The images used in this report were collected from the search engine DuckDuckGo using the search terms ‘Christmas jumper’ and ‘Jumper’ to create the festive and general classes respectively. Additional search filters, image_type=transparent, and image_layout=square, were utilised to ensure a uniform dataset. Images were collected using the jmd-imagescraper python package(Dockrill, 2020).

To be included in the final dataset, images must include one jumper on a pure white background. Criteria for exclusion include: cartoon or blurry images, other clothing items and images containing multiple jumpers, see Figure 1.

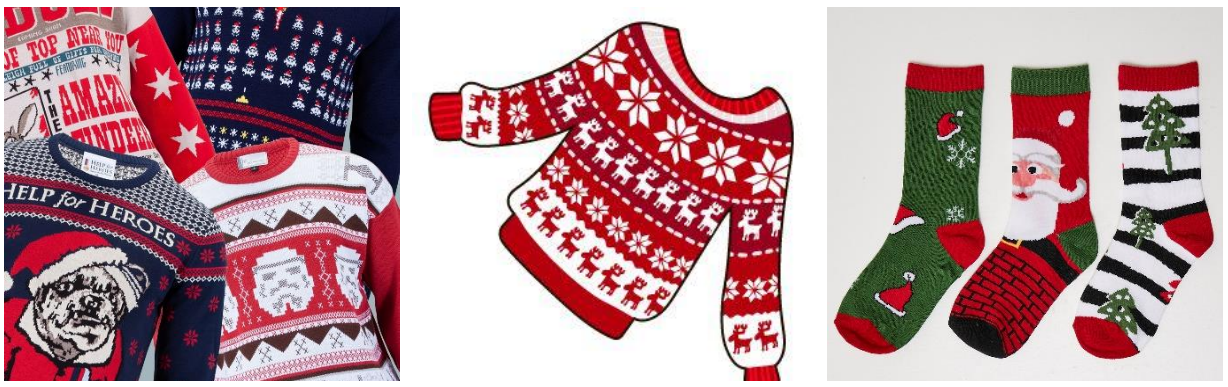


Figure 1: A selection of inappropriate images removed from the dataset

The final dataset included 473 Christmas jumpers and 561 general jumpers, a sample of which is shown in Figure 2.

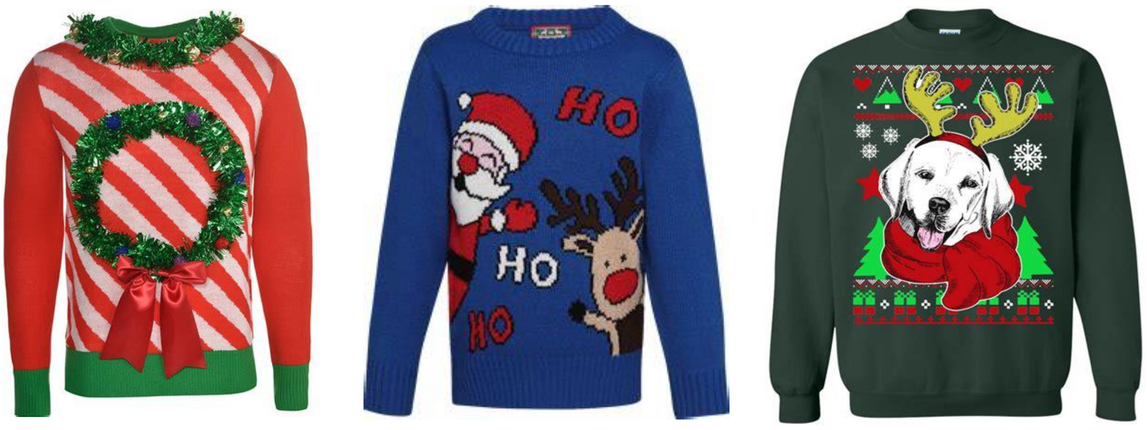


Figure 2: A Selection of images from the Christmas jumper class included in the final dataset

Data Exploration

Data Exploration shown in Figure 3 reveals a range of both sizes and shapes.

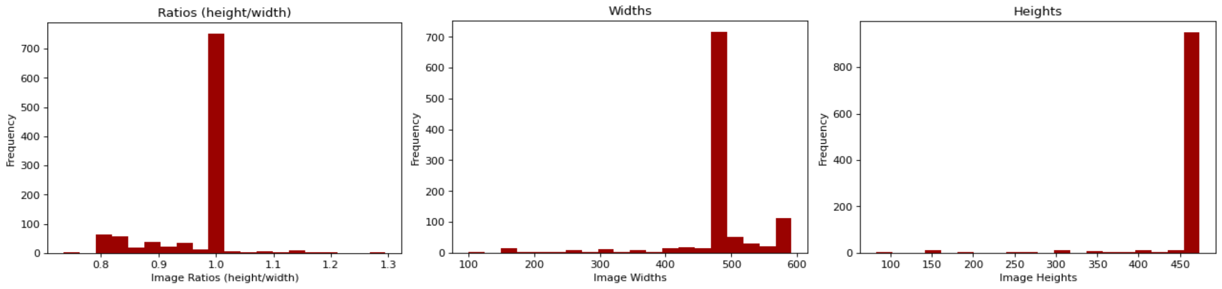


Figure 3: Histograms to show the distribution of shapes(L), Widths(Centre) and Heights (R)

Pixel averages were plotted in position to explore variation in images. Figure 4 shows less variation in the Christmas class than in the base class. This is expected since Christmas Jumpers is a subcategory. However, the bright label and central square in the Christmas jumper overlays suggest the dataset includes highly-similar images that may interfere with the model.

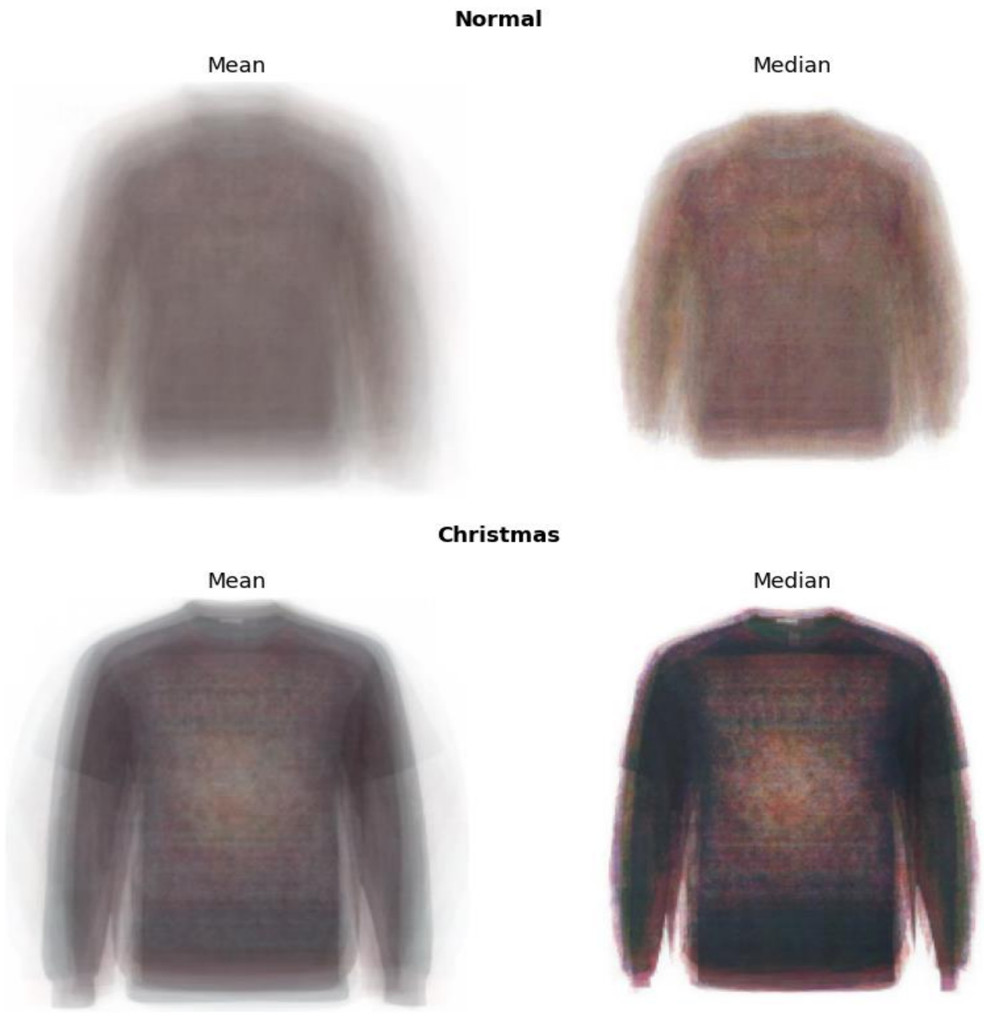


Figure 4: Pixel averages for the training set of each class

Dataset limitations

This dataset might be limited in utility due to overly consistent images. The model may perform poorly with user images due to low lighting or busy backgrounds.

In creating this dataset, highly ambiguous images were removed, see Figure 5. This process introduced bias as it was done by one person. To reduce this bias in future experiments, classes should be cleaned by a group of diverse individuals.

Figure 5: A divisive jumper that may be classified as 'Christmas' or 'Winter' dependent on the individual

Methods

Convolutional Neural Networks

Traditional Deep Neural Networks (DNNs) consist of fully connected layers of neurons activated by activation functions.

DNNs can be used for small images by converting them to a 1-dimensional array. However, this is unmanageable for larger images. For the 224x224 pixel images used in this report, with a 1000-neuron input layer (this is already very restrictive) there would be over 50 million connections in the first layer alone.

The preferred network type for computer vision is a Convolutional Neural Network (CNN). A CNN consists of feature extraction and fully connected layers. Feature extraction layers include convolutional layers, de-linearising layers and pooling layers, explored in the following sections.

Convolutional Layers

In a convolutional layer, filters move across the image and calculate the dot product of itself with the pixel subset below (Goodfellow et al., 2016).

These values create a new matrix called a Feature Map. Different kernels can be used to detect edges, vertical and horizontal lines, circles, and corners. Later stages of the network develop more complex kernels that can detect combinations of these shapes (LeCun et al., 2015).

Convolutional layers result in feature maps that are smaller than the original image, see Figure 6. Since the kernels are small, lost data is usually negligible however after successive convolutional layers it can compound. To maintain the original dimensions, the image is expanded by empty pixels, called padding (Zhang et al., 2021).

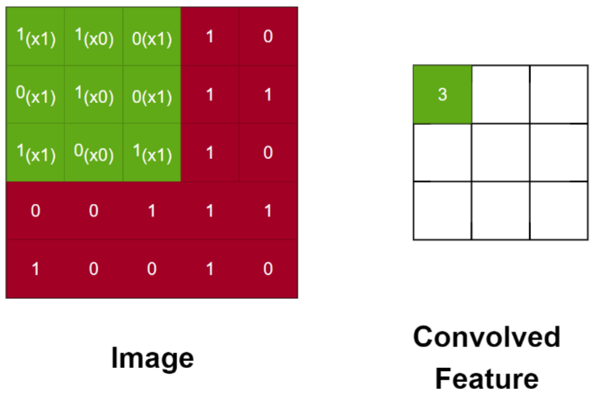


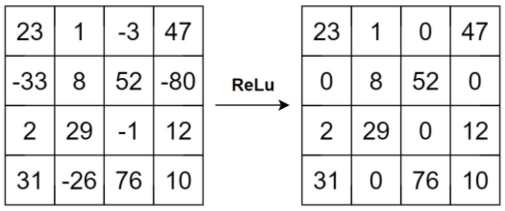
Figure 6: Shows the process of a convolutional layer

Activation Functions

A Rectified Linear Unit (ReLu) is a non-linear activation function that removes negative values from the feature map to combat the vanishing gradient problem (Hochreiter, 1998; Nair and Hinton, 2010). More information on the vanishing gradients problem can be found in Appendix B. ReLu is the preferred activation function for hidden layers, with sigmoid used for the output layer of a binary classification problem (as explored in this report) (He et al., 2016).

$$f(x) = \max(0, x)$$

Equation 1: ReLu function (Nair and Hinton, 2010)



23	1	-3	47
-33	8	52	-80
2	29	-1	12
31	-26	76	10

23	1	0	47
0	8	52	0
2	29	0	12
31	0	76	10

Figure 7: An example input and output of a ReLu function

Pooling Layers

Pooling layers reduce the dimensionality of feature maps to lessen the computational load and reduce overfitting. A pooling layer aims to merge semantically similar features by returning a single value for each subset of pixels. Max Pooling is the preferred type of pooling layer. It returns the maximum value in each segment of the data, keeping only the most prominent features and removing the noise (Zafar et al., 2022).

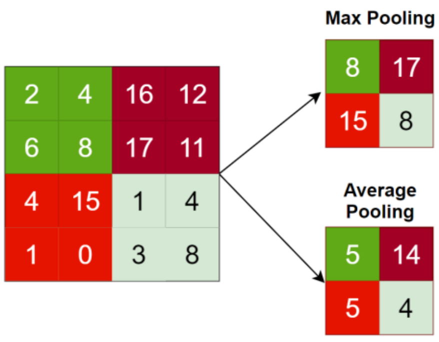


Figure 8: Comparing the output of Max Pooling and Average Pooling

Pooling layers introduce invariance into the model because minor translations will produce the same pool, see Figure 9 (Aurélien Géron, 2019).

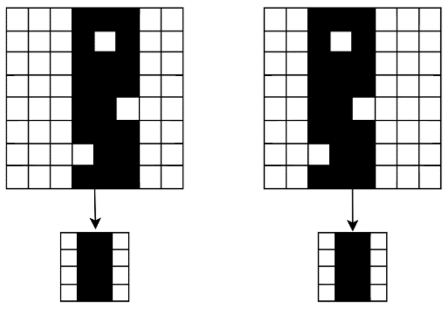


Figure 9: Compares the output of Max Pooling of translated images with a 2x2 filter

Drop-out Layers

Drop-out layers randomly select a given percentage of neurons in the fully connected layers to reset to zero, preventing overfitting. When overfitting occurs neurons can develop complex co-adaptions by using a later node to correct the output of previous nodes. Dropout layers prevent this by making the presence of the previous node unreliable (Srivastava et al., 2014).

Limitations of CNN's

A problem for the implementation of CNNs is the requirements for large training datasets, for small datasets training a CNN from scratch can be unfeasible. The ability of CNNs to identify an object regardless of its location in an image is one of its major strengths. However, due to the image segmentation that allows this, the spatial interaction of objects is lost. If this is essential to the problem at hand, CNNs will not perform well (Hosseini et al., 2017).

Despite this, they are the industry standard for image classification problems and will be the class of artificial neural network used throughout this report.

Performance evaluation metrics

Accuracy is the number of cases correctly classified in a test set. This is the simplest and most used performance metric. However, it is only valid when the number of examples in each class are equal (Luque et al., 2019). Since the classes are approximately equal this will be the primary evaluation metric used in this report.

$$accuracy = \frac{number\ of\ correctly\ classified\ cases}{total\ number\ of\ cases}$$

Equation 2: Accuracy (Petridis, 2018)

Additionally, a confusion matrix will also be used to evaluate model performance. A confusion matrix is a contingency table that compares actual and predicted values (Prateek, 2017).

		Predicted	
		Positive	Negative
Actual	Positive	True Positives	False Negatives (Type II error)
	Negative	False Positives (Type I error)	True Negatives

Figure 10: Confusion Matrix

A false negative in the Christmas jumper classification problem is when a Christmas jumper is identified to be a regular jumper and would result in inappropriate outfit recommendations.

A false positive is a regular jumper being identified as festive; this would result in a jumper being withheld from recommendations for a large portion of the year.

A false positive may be more severe due to the lack of visibility of the error since it's much easier to flag an inappropriate recommendation than to notice a jumper's absence in recommendations. This is of course dependent if flagging is enabled in the styling app. Since this is unknown, percentage accuracy and confusion matrices will be used to evaluate models in this report.

Experimental setup

Data Pre-processing

To enable the images to be used as inputs for a CNN they were first resized, so the biggest dimension was 224 pixels and then padded to make them square, see Figure 11.

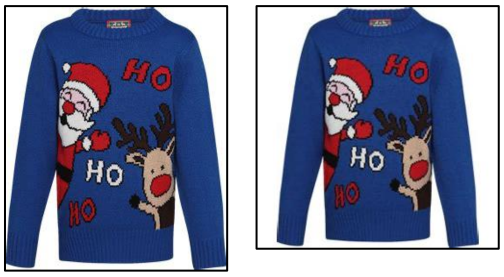


Figure 11: an example of resizing and padding.
Original (left) 252x300, transformed (right) 224x224

Then, images were normalised to reduce sensitivity to small changes in pixel values and enable quicker convergence. An example of this can be seen in Figure 12.

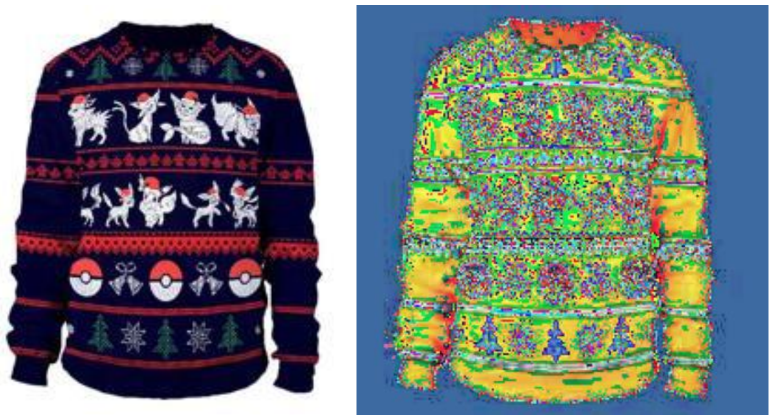


Figure 12: Showing the before (L) and after (R) of image normalisation

Finally, the dataset was split into train, test, and validate sets with a ratio of 5:3:2 respectively.

Data Augmentation techniques such as rotation, image flipping, or cropping could have been introduced to increase the size and variation of the dataset. However, since this would not be representative of the data users would upload this was not included within the scope of this project.

Models

In the pursuit of finding the most accurate model for classifying Christmas jumpers, both pre-trained and original models were tested.

Pre-trained networks

As mentioned previously, the primary layers of a CNN identify very simple shapes such as horizontal and vertical lines and more complex objects are detected as it goes deeper.

To avoid repeating this computation we can use the weights and biases from pre-trained networks. This reduces computational load and improves accuracy as pre-trained networks were trained on a much larger dataset than is available for this study (Nogueira et al., 2016).

There are two ways to use pre-trained models: finetuning or using them as feature extractors. Finetuning initialises all weights and biases with pre-trained values and then adjusts based on the new data. Using the network as a fixed feature extractor only changes the weights and biases in the fully connected layers. In both methods, the final layer needs to be changed to a binary classification. Although computationally more expensive, fine-tuning usually gives better results (Hadhrami et al., 2018).

This report will use finetuning with the following pre-trained models, VGG, GoogLeNet and AlexNet. Their architectures are explained in Appendix A.

Custom built CNN's

Base models were created to compare to the intricate architectures designed for the ILSVRC. These custom CNNs architectures can be found in Figure 13.

Models 1-5 experiment with changing the depth of the network while maintaining structure. Model 6, compared with model 1, explores the effect of consecutive fully connected layers. Model 7 is designed to test the effect of introducing complexity, inspired by the VGG architecture shown in Appendix A.

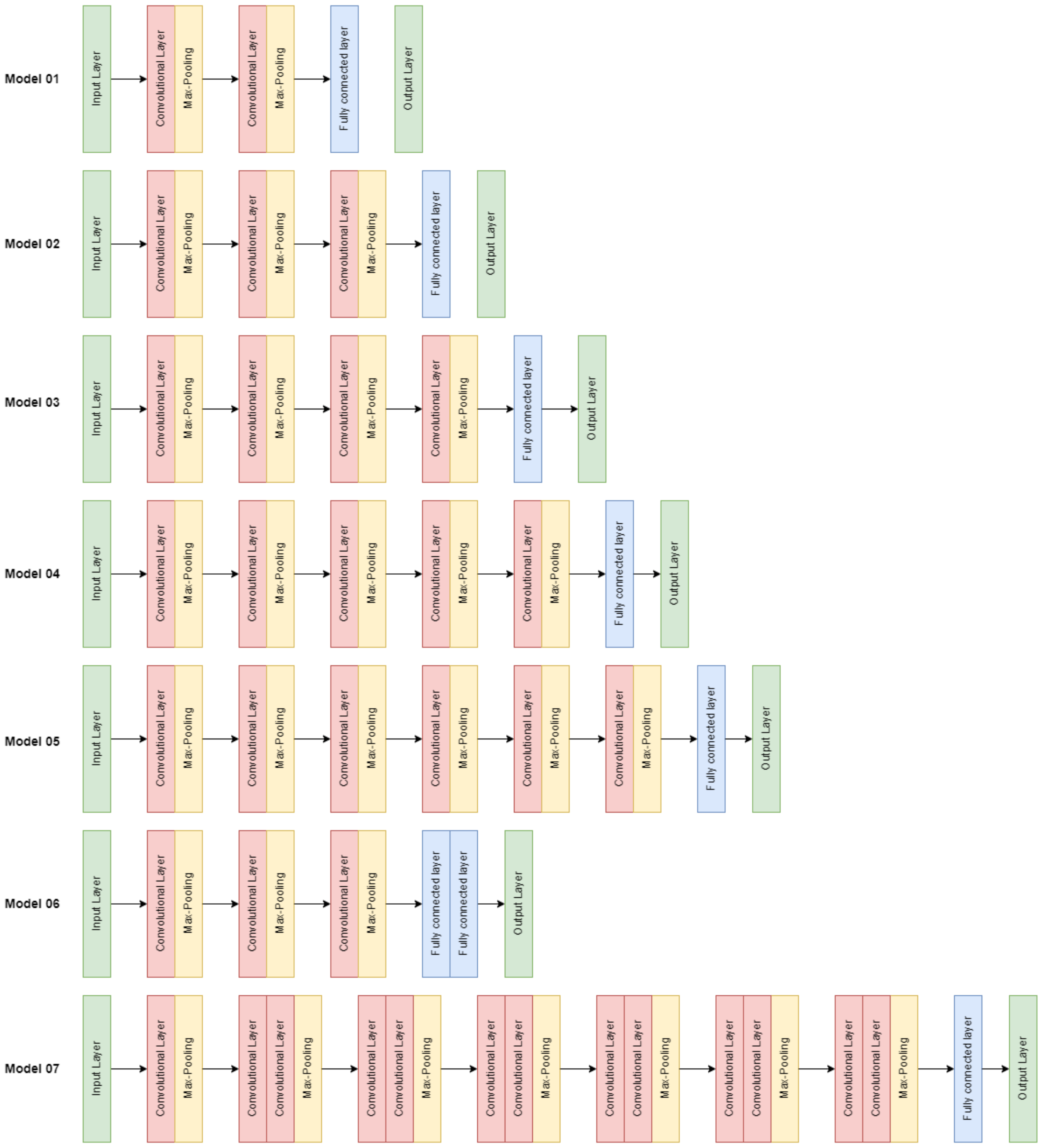


Figure 13: Custom CNN architectures

Results and Discussion

Custom CNN's

Models 1-5 performed equally regardless of the number of convolutional layers.

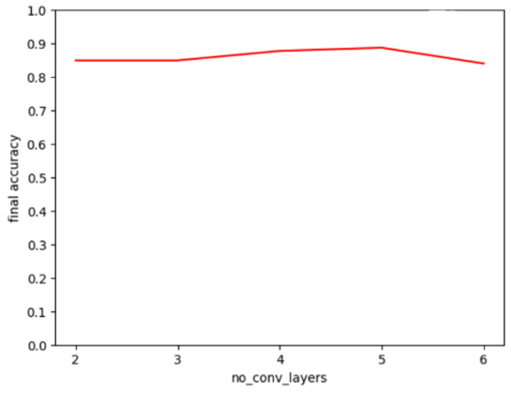


Figure 14: Shows accuracy plotted against the number of convolutional layers in models 1-5

This result may be due to too high a learning rate that leads to extreme overfitting. To test this hypothesis, model 1 was trained with a range of learning rates. This did not fix the overfitting, and the accuracy remained constant. The results of this experiment can be seen in Appendix C - Changing Learning rate.

This suggests that the model is identifying the classes by a simpler metric. For example, in 2019, a neural network learnt to classify a 'horse' using a tag stating the image source present in one-fifth of horse images (Lapuschkin et al., 2019). In this case, either an invisible watermark could be present due to the unknown data sources, or the model could be using colour or shape. Alternatively, data leakage, when data outside of the training set is used to create the model, could be occurring.

Model 6, was tested to determine the effect of consecutive fully connected layers. The model was significantly overfitting, as can be seen in Figure 15.

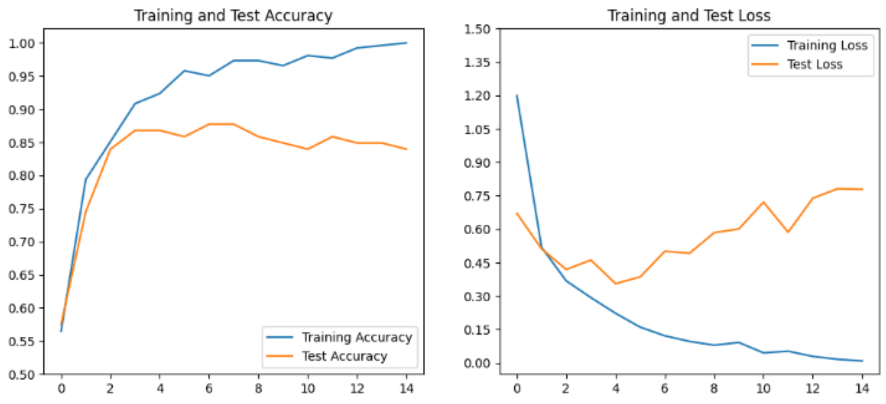


Figure 15:An accuracy loss graph showing model 6 overfitting without dropout layer

A dropout layer with a probability of 0.5 was introduced. This reduced overfitting and brought the accuracy in line with models 1-5 (86.1%). To maintain the model's simplicity, model 6 was discarded at this stage.

Model 7 achieved a similar accuracy (84.7%) and was disregarded for the same reason.

Greyscale dataset

To test the theory that the models were differentiating based on colour, models 1-5 were run with a greyscale version of the dataset.

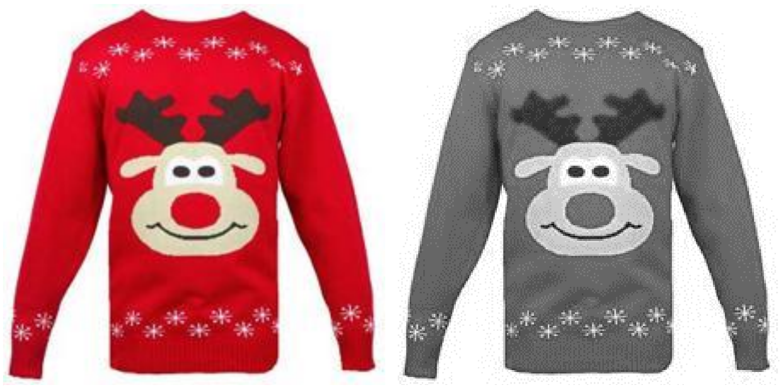


Figure 16: Transforming images to greyscale

Converting the images to greyscale both improved accuracy and decreased overfitting, see Figure 17.

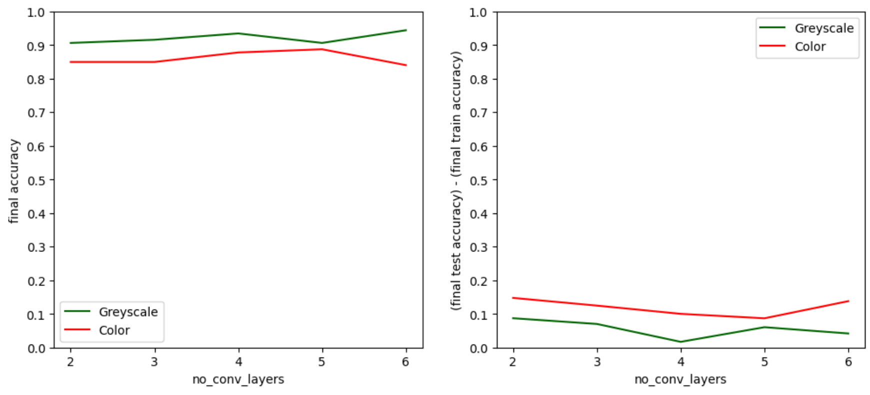


Figure 17: Compares the results of custom CNNs trained on coloured and greyscale images

However, the shallow neural network with 2 convolutional layers still performed better than expected suggesting the lack of variation seen in "Figure 4: Pixel averages for the training set of each class" may be the source of the data leakage.

Pretrained Models

The AlexNet and VGG overfit the coloured dataset, which can be seen by the distance between the training and test lines in their respective loss graphs in Figure 18. They also stop learning in epoch 4. This is likely because of the limited dataset provided.

As expected by the ILSVRC results, the GoogLeNet was the most accurate model. Overfitting occurs when a network gets too deep and has too many unrestricted variables and GoogLeNet avoids this because its inception modules make the model wider rather than deeper, as can be seen by in architecture in Appendix A.

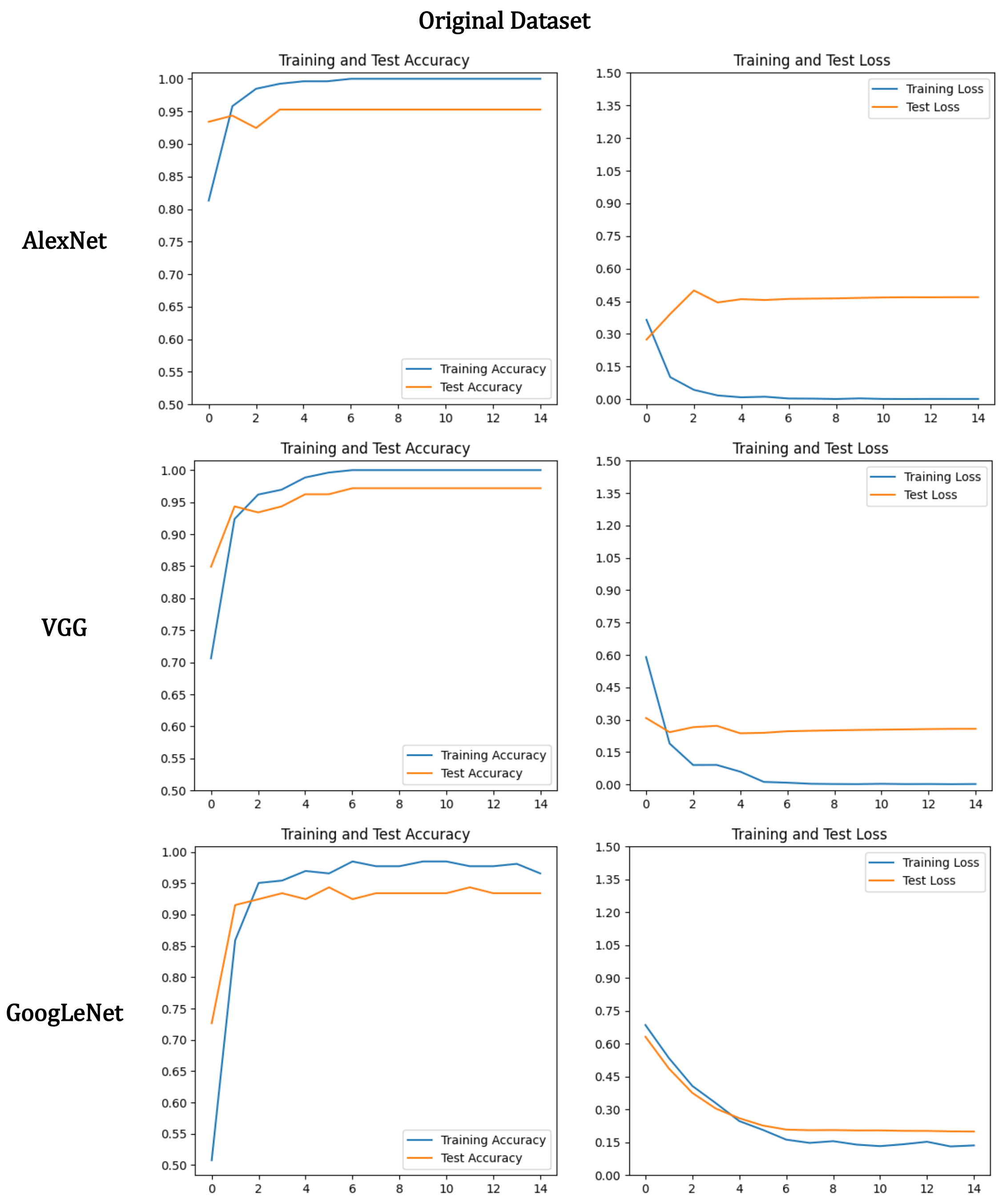


Figure 18: Accuracy loss graphs for AlexNet, VGG, and GoogLeNet trained on the coloured dataset

Greyscale dataset

The greyscale dataset reduced overfitting in all 3 pre-trained models, as is especially evident when comparing the loss graphs in Figure 18 to Figure 19. The AlexNet and VGG are still overfitting because their final test accuracy is 1.00.

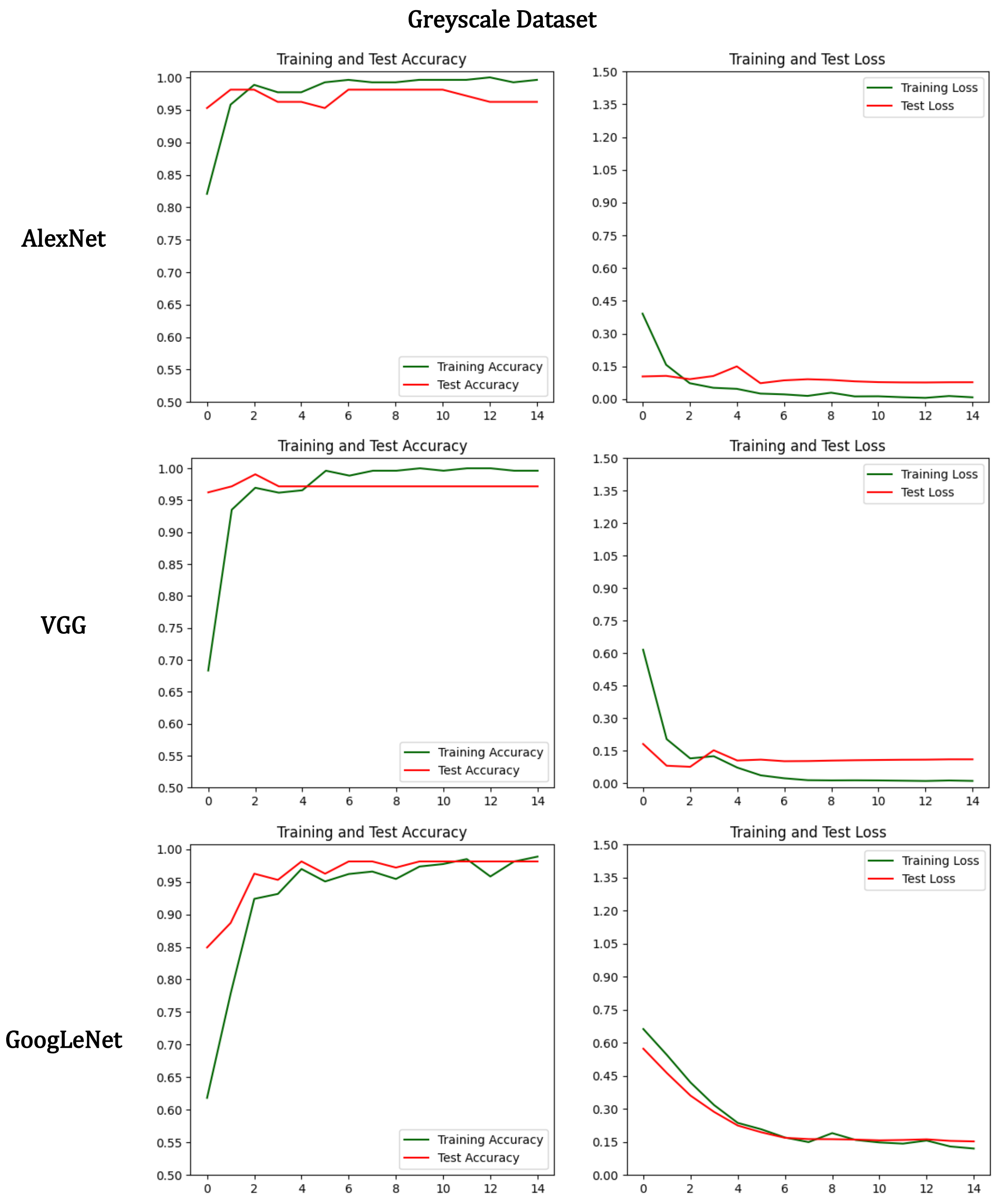


Figure 19: Accuracy loss graphs for AlexNet, VGG, and GoogLeNet trained on the coloured dataset

Reducing Learning rates

To reduce overfitting in AlexNet and VGG the learning rate was reduced from 0.001 to 0.0005. This had minimal effect on AlexNet but did reduce overfitting in the VGG network as can be seen by its final train accuracy in Figure 20. Reducing the learning rate negatively impacted the accuracy of GoogLeNet.

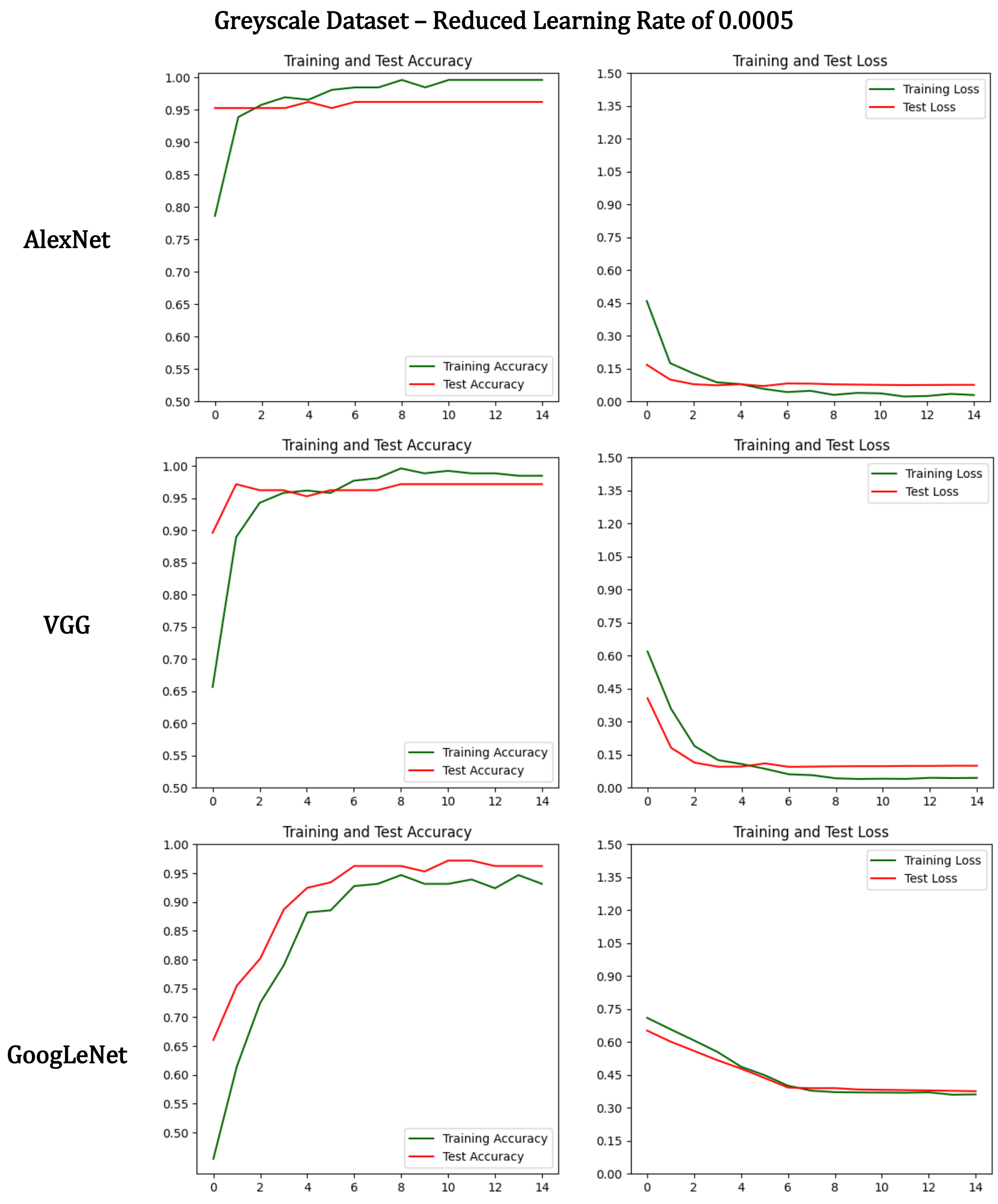


Figure 20: Accuracy loss graphs for AlexNet, VGG, and GoogLeNet trained on the greyscale dataset with a learning rate of 0.0005

Final Model

This report aims to find a model that can successfully classify Christmas jumpers from normal jumpers and to that end, the final model is the GoogLeNet architecture finetuned with a greyscale Christmas jumper dataset and a learning rate of 0.001. When run with the validation set it achieved an accuracy of 93.67, which is broken down in the confusion matrix in Figure 21.

		Predicted	
		Christmas	Normal
Actual	Christmas	71	3
	Normal	7	77

Figure 21: Confusion matrix of results

Figure 22 shows incorrectly classified images in their greyscale state. These images appear ambiguous, and this suggests they should have been removed from the dataset based on the criteria stated in "Dataset creation".

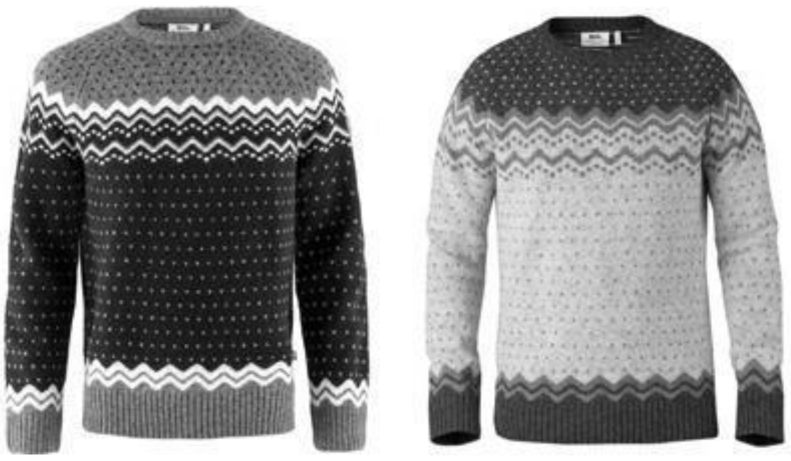


Figure 22: Incorrectly classified images sown in greyscale

However, by reverting them to their coloured state in Figure 23 it is clear that they are winter jumpers. A solution for this is to create guidance for data labellers and to ensure the data labellers are viewing the images in their final transformed state, i.e., the greyscale images shown in Figure 22.

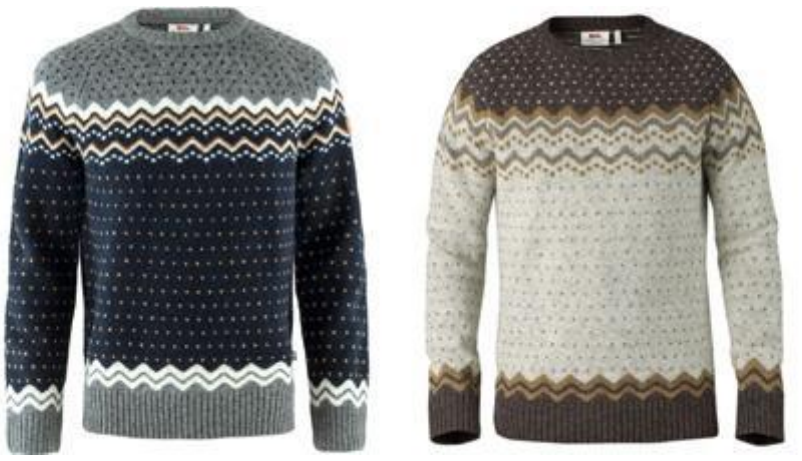


Figure 23: Incorrectly classified images shown in colour

Conclusion

Recommendations for future study

Future researchers are encouraged to develop a more robust dataset to re-evaluate the techniques in this report. This could include data augmentation such as rotation, scaling and adding and removing noise to increase the size of the dataset and more closely align with real user data. Additionally, a realistic dataset could be created from images submitted by users to a styling app.

Additional research could include finetuning a pre-trained CNN that has been trained on more similar input data. The input must be similar such that classification is based on the pattern of an object present in the image rather than the shape of the object itself.

Ethical/Legal/Professional considerations

The use of this model in styling apps may become frustrating to non-Christmas celebrators. They may not want their clothing to go through the model and be frustrated if some winter jumpers are only recommended in December. To avoid this, allowing the user to toggle the inclusion of the model on and off may be beneficial.

A professional and legal consideration would be whether to use future user-inputted data to improve the model over time. Using this data would drastically improve the quality of the model as it is more relevant than the test sets created from web scraping. However, if implementing this, data would have to be collected, stored, and cleaned securely and users would need to be aware of how this data was being used.

Closing words

This report aimed to create a model capable of classifying Christmas jumpers to improve outfit recommendations in styling apps. The final model achieved an accuracy of 93% which satisfies this aim. From the incorrectly classified images, it is evident that the dataset used is the limiting factor for the model and with a more robust dataset the accuracy could be expected to improve significantly.

References

Aurélien Géron, 2019. Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems, 2nd Edition. ed, O'Reilly Media. O'Reilly Media Inc.

Chen, S.H., Jakeman, A.J., Norton, J.P., 2008. Math Comput Simul 78, 379–400.

Dawkins, P., 2023. Step Functions [WWW Document]. Lamar University. URL https://tutorial.math.lamar.edu/classes/de/StepFunctions.aspx (accessed 1.17.23).

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., 2009.

Dockrill, J., 2020. jmd-imagescraper Python Package.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press.

Hadhrami, E. al, Mufti, M. al, Taha, B., Werghi, N., 2018. 2018 International Conference on Artificial Intelligence and Big Data, ICAIBD 2018 148–154.

He, K., Zhang, X., Ren, S., Sun, J., 2016. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2016-December, 770–778.

Hochreiter, S., 1998. International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems 6, 107–116.

Hosseini, H., Xiao, B., Jaiswal, M., Poovendran, R., 2017. University of Washington.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012.

Lapuschkin, S., Wäldchen, S., Binder, A., Montavon, G., Samek, W., Müller, K.-R., 2019. Nat Commun 10.

LeCun, Y., Yoshua, B., Geoffrey, H., 2015. Nature 521, 436–444.

Luque, A., Carrasco, A., Martín, A., de las Heras, A., 2019. Pattern Recognit 91, 216–231.

Nair, V., Hinton, G.E., 2010. Department of Computer Science, University of Toronto.

Nogueira, K., Penatti, O.A.B., Santos, J.A. dos, 2016. Pattern Recognit 61, 539–556.

Petridis, S., 2018. Evaluating Hypothesis - Machine Learning (course 395). Imperial College London.

Prateek, J., 2017. Artificial intelligence with Python : build real-world artificial intelligence applications with Python to intelligently interact with the world around you. Packt Publishing.

Rabbit Tech Inc., 2023. Smart Closet: Your Personal Stylist.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L., 2015a. Int J Comput Vis 115, 211–252.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C, Fei-Fei, Li, Russakovsky, O, Deng, J, Su, H, Krause, J, Satheesh, S, Ma, S, Huang, Z, Karpathy, A, Khosla, A, Bernstein, M, Berg, A C, Fei-Fei, L, 2015b.

Simonyan, K., Zisserman, A., 2015.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. The Journal of Machine Learning Research 15, 1929–1958.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 07-12-June-2015, 1–9.

Xiao, H., Rasul, K., Vollgraf, R., 2017.

Zafar, A., Aamir, M., Mohd Nawi, N., Arshad, A., Riaz, S., Alruban, A., Dutta, A.K., Almotairi, S., 2022. Applied Sciences 2022, Vol. 12, Page 8643 12, 8643.

Zhang, A., Lipton, Z.C., Li, M., Smola, A.J., 2021. arXiv preprint arXiv:2106.11342.

Appendices

Appendix A – Pretrained Architectures

AlexNet

The AlexNet architecture consists of 5 convolutional layers, 3 max-pooling layers and 2 fully connected layers the arrangement of which can be seen below. Each convolutional layer consists of the filter application and the ReLu activation function. Filter sizes of 11x11, 5x5, and 3x3 are used with strides of 4, 1, and 1, respectively, reducing deeper into the network.

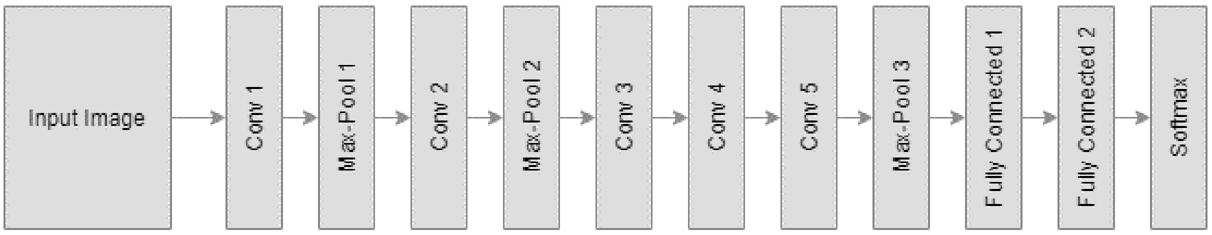


Image used with permission from Geeksforgeeks.org

The pooling has a filter size of 3x3 and a stride of 2 pixels.

Input size is fixed because of the fixed nature of the fully connected layers. The input size is 224x224x3 for images but 227X227X3 with padding.

Between the fully connected layers is a dropout layer with probability 0.5.

Soft max is used in the output layer and is used in the subsequent VGG and GoogLeNet models also. The Soft max function transforms all input values to a value between 0 and 1 so they can be interpreted as probabilities and the output with the highest probability is given as the prediction.

AlexNet overall has over 60 million parameters

VGG

The VGG-16 consists of 5 stacks of convolutional layers and Relu functions, separated by pooling layers and finishing with 3 fully connected layers. VGG uses 3x3 filters with a stride of 1 which is much smaller than previous networks (for example AlexNet has a receptive field of 11x11 with a stride of 4).

Combining stacks of convolutional layers allows the network to imitate a larger receptive field (for example the first stack has two layers with 3x3 filters which mimic a 5x5 receptive field. The benefit of combining small layers is to reduce the number of parameters and increase the number of ReLu layers thus allowing the network to converge quicker.

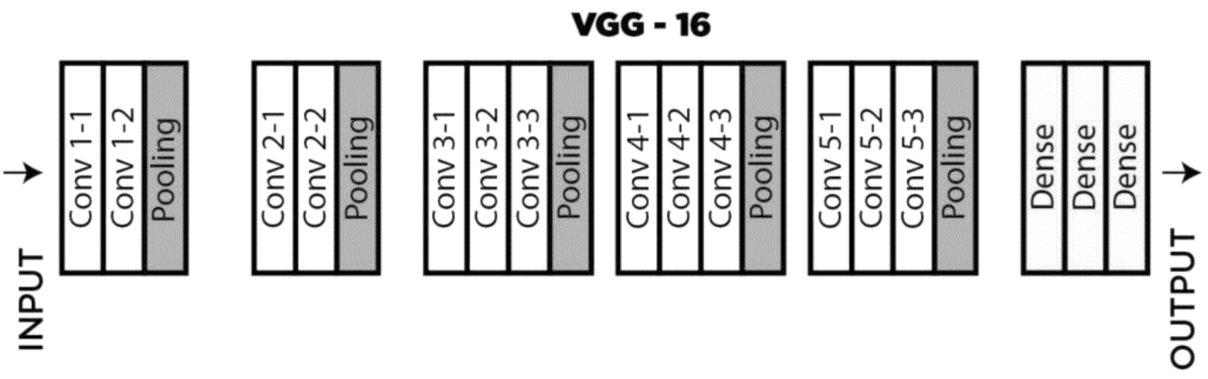


Image used with permission from Geeksforgeeks.org

The pooling has a filter size of 2 x 2 and a stride of 2 pixels, halving the size of the activations.

The VGG has over 138M parameters.

GoogLeNet

The GoogLeNet architecture starts with 2 convolutional layers divided by max-pooling layers. It then consists of 3 blocks of inception modules separated by max pooling layers. It concludes with a dropout layer (dropout ratio=0.7) and a single fully connected layer and softmax classifier to provide the output.

The GoogLeNet inception module, several convolutional filters with different sized filters and a max-pooling layer are applied to in parallel and the results are concatenated. To enable this concatenation the feature maps must be the same size so padding is added to ensure uniformity.

To prevent the computational load from becoming unmanageable, 1x1 convolutions are used to reduce the depth of the feature maps while preserving the x-y information.

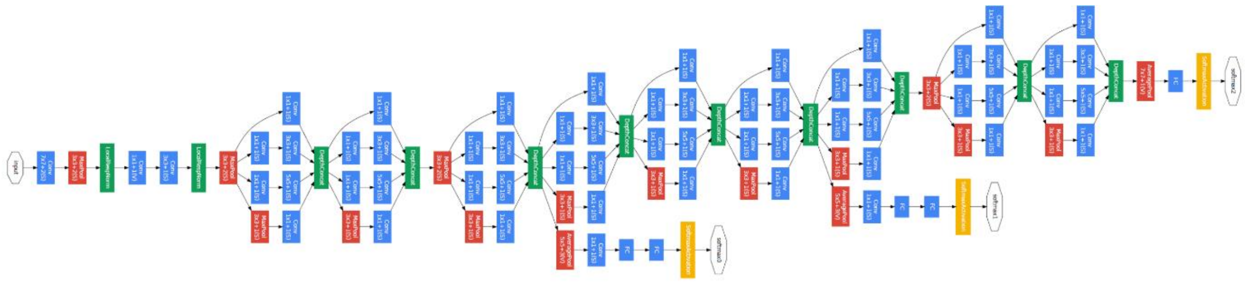


Image used with permission from googleblog.com

The pooling has a filter size of 5x5 and a stride of 3 pixels.

Appendix B – Activation functions and the vanishing gradients problem

Simple perceptron's use a step function. The output is 1 or 0 depending on the sign of the summed activation of the node[1].

$$f(x) = \begin{cases} 0 \; if \; 0 > x \\ 1 \; if \; x \geq 0 \end{cases}$$

$\boldsymbol{Step\ function}$ (Dawkins, 2023)

Since this loses a lot of information, DNNs use sigmoid neurons with a sigmoid activation function which produces a value between 0 and 1.

$$\sigma(x) = \frac{1}{1 + e^{-\beta x}}, \qquad \beta > 0$$

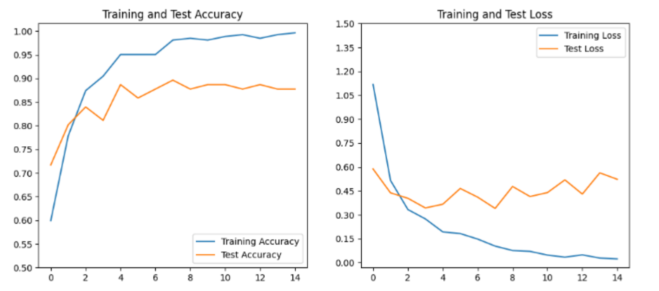
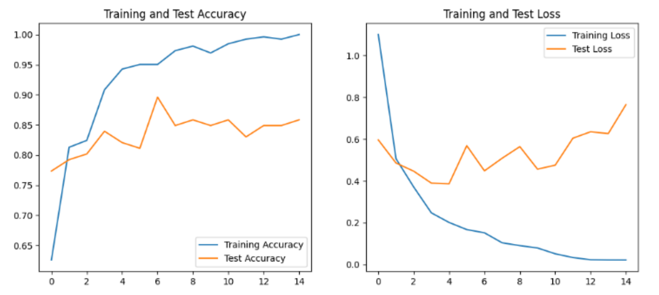
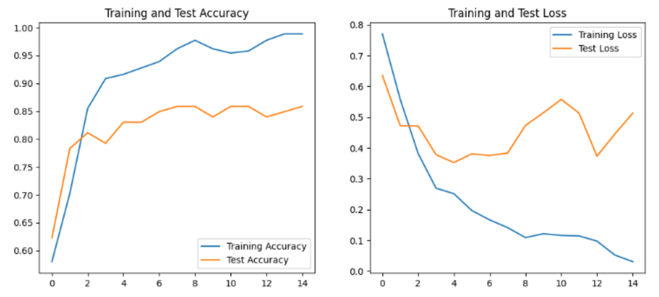
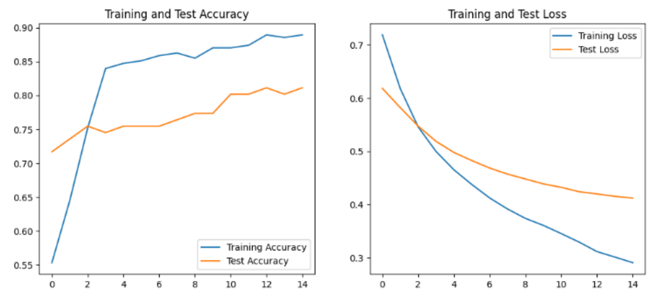
$\boldsymbol{Sigmoid\ function}$ (Chen et al., 2008)

The sigmoid function is only sensitive to changes close to the midpoint, high and low values are pushed to 1 and 0 respectively. Furthermore, when the sigmoid function has been applied repeatedly during backpropagation, the gradient diminishes and can become negligible close to the input layer. It then becomes impossible to effectively adjust the weights and biases in these early layers – this is the 'vanishing gradients' problem.

[1] The summed activation of the node is the sum of the weights and biases.

Appendix C – Optimisation outputs

1. Changing Learning rate

Learning Rate	Accuracy/Loss Graphs
0.001	
0.0005	
0.0002	
0.00002	

Appendix D – GitHub Repository

https://github.com/gvaughan2000/classifying_christmas_jumpers