Assignment 2

Stent, Mark

CETM26

Wordcount: 3216

Table of Contents

1. Figure list

2. Introduction & Background

'If We Want Machines to Think, We Need to Teach Them to See.'-Fei Fei Li

Computer Vision (CV) is the most popular Artificial Intelligence (AI) technique found in AI applications around us, used in various applications, including facial recognition, self-driving cars, augmented reality, etc. The interest in having deeper hidden layers has recently surpassed classical methods performance in different fields, especially pattern recognition (Albawi, Mohammed and Al-Zawi, 2017). The convolution neural network (CNN) created in recent years has evolved into a robust and widely used deep learning model. It has been employed in image processing as it effectively deals with image classification and recognition problems and has improved the accuracy of many machine learning tasks.

This paper will be using CNNs and various other optimisation techniques to explore and experiment with the CIFAR 10 image dataset collected and labelled by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton.

Section two starts by exploring the dataset and performing some transformations. Section three then explains the experimental rationale of each process iteration, which includes creating a baseline architecture, adding and subtracting various Keras Sequential layers, hyperparameter tuning and a stretch experiment with transfer learning. Section four shows a series of graphs and tables to plot out the results. Section five evaluates these results and describes my experiences and learning points through this set of experiments. Finally, section six discusses the pros and

cons of computer vision in an ethical context.

3. Methodology & Experimental design

There is currently no formal method for creating deep learning models. The large permutations of possible parameters and hyperparameters make it hard to develop a tool that can accurately determine whether a model is correct (Urban and Miné, 2021); this, coupled with the enormity of computing power needed to solve these problems with massive volumes of data, make the problem of image classification a challenging one.

To perform my experiments for this project, I used Kaggle.com and Google Collab notebooks as they offer free limited GPU (graphics processing units) services. GPUs allow deep network models to be processed parallelly and thus speed up model training time.

I broke the experiment process down into the following tasks:

1. Data preparation and exploration
2. Create a basic bassline architecture with constant hyperparameters to compare against potentially improved architectures.
3. Find an optimal architecture by adding or subtracting various layers.
4. Tune the hyperparameters of this model.
5. Experiment with transfer learning to try and increase the strength of the model

3.1.Data preparation and exploration

I began by importing the relevant libraries needed complete each of the tasks (Appendix 1). Then, I loaded the dataset from the Tensorflow Keras datasets module. The 60000 images were then split up into 40000 training images, 10000 validation images, and 10000 hold out test images to be used to evaluate the model. Sample images are shown in Figure 1.

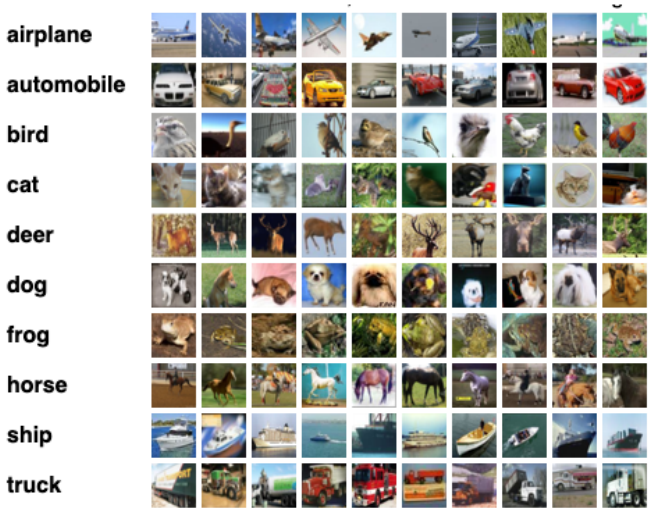

Figure 1- Cifar 10 Sample data

The distributions of the 10 class variables are approximately balanced (Figure 2), which means that 'accuracy' is a relevant metric for this multiclass classification problem (neptune.ai, 2021).

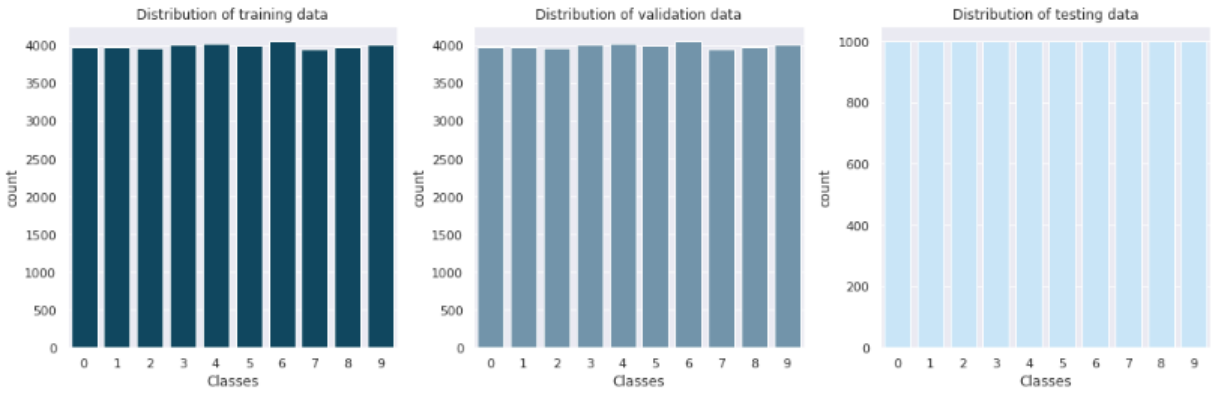

Figure 2 - Dataset distribution

I then normalized each image to ensure each pixel had a similar distribution and then one-hot encoded the target variables, enabling the models to converge faster (Nikhil B, 2017).

3.2.Baseline architecture

According to Nair (2022), "a baseline model is essentially a simple model that acts as a reference in a machine learning project. Its main function is to contextualize the results of trained models."

To make the experiment process more effective, I created a function that accepted arguments to build and compile Sequential Keras layers, including Conv2D, Max pooling, Dense and Dropout layers, into a model. This function held the following hyperparameters as constant:

- Epochs: 20 (for computational reasons, this was kept small)
- Activation function: Relu (it is fast and works well for general situations)
- Dropout rate: 0

- Output activation function: Softmax (this is a multiclass classification problem with ten outputs)
- Optimizer: ADAM with 0.001 learning rate (Adam has fewer hyperparameters and has a fast computation time)
- Conv2d filters: 32
- Dense filters: 128

The baseline architecture included one Conv2D input layer, one Conv2D layer, 1 Max-pooling layer, one Flatten layer and one Dense output layer (Figure 3).

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 32, 32, 32)        9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 10)                81930
=================================================================
Total params: 92,074
Trainable params: 92,074
Non-trainable params: 0
_________________________________________________________________
```

Figure 3 - Baseline architecture

3.3.Adding and subtracting Sequential layers

The next task was to iteratively add layers to the baseline model and track the improvements or deteriorations. I tried multiple Conv2D layer configurations, Dropout layers, Dense layers and Max Pooling layers and recorded the results. Next, I added data augmentation to the model to

increase the generalization and diversity of the training set (Data augmentation, 2022). For the augmentation, I added a random rotation range of 15 degrees, a width and height shift range of 0.1 and a horizontal flip. Figure 4 shows the various models used to iteratively better base architecture.

	Model
0	Baseline Model: 2 Conv, 1 max pool, 1 dense output
1	Model I: 4 Conv, 2 max pool, 1 dense output
2	Model 2: 6 Conv, 3 max pool, 1 dense output
3	Model 3: 6 Conv, 3 max pool, 1, dense layer, 1 dense output
4	Model 4: 6 Conv, 3 max pool, 1, dropout, 1 dense output
5	Model 5: 6 Conv, 3 max pool, 1, dense layer, 1, dropout, 1 dense output, batch norm between layers
6	Model 6: 6 Conv, 3 max pool, 1, dense layer, 1, dropout, 1 dense output, batch norm between layers, 50 epochs
7	Model 7: Model 5 with data augmentation

Figure 4 - Model architectures

3.4.Hyperparameter tuning

I used the Keras Tuner RandomSearch to optimize the filter sizes of each Conv2D layer to see which features were important in generalizing; I started at 16 and tested in steps of 16 up to a maximum of 64. I also optimized the dropout percentage parameter, ranging from 0 to 0.5 in steps of 0.1; Dropout stops neurons from synchronously optimizing their weights (O'reilly, 2022), thus, it helps in training and assists in preventing overfitting. Furthermore, tuning the learning rate is a vital hyperparameter. One that is too small can cause the gradient descent process to get stuck, and too large can cause the model to converge to a suboptimal solution (https://www.facebook.com/MachineLearningMastery, 2019); values ranging from 0.02 to 0.004 were tested. Finally, I tried the Relu and Elu activation functions as hyperparameters on the Conv2D Layers. The final model for this group of experiments shown in Figure 5.

```
Model: "sequential_6"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_30 (Conv2D)           (None, 32, 32, 32)        896

batch_normalization_6 (Batc  (None, 32, 32, 32)        128
hNormalization)

conv2d_31 (Conv2D)           (None, 32, 32, 32)        9248

batch_normalization_7 (Batc  (None, 32, 32, 32)        128
hNormalization)

max_pooling2d_15 (MaxPoolin  (None, 16, 16, 32)        0
g2D)

conv2d_32 (Conv2D)           (None, 16, 16, 32)        9248

batch_normalization_8 (Batc  (None, 16, 16, 32)        128
hNormalization)

conv2d_33 (Conv2D)           (None, 16, 16, 32)        9248

batch_normalization_9 (Batc  (None, 16, 16, 32)        128
hNormalization)

max_pooling2d_16 (MaxPoolin  (None, 8, 8, 32)          0
g2D)

conv2d_34 (Conv2D)           (None, 8, 8, 32)          9248

batch_normalization_10 (Bat  (None, 8, 8, 32)          128
chNormalization)

conv2d_35 (Conv2D)           (None, 8, 8, 32)          9248

batch_normalization_11 (Bat  (None, 8, 8, 32)          128
chNormalization)

max_pooling2d_17 (MaxPoolin  (None, 4, 4, 32)          0
g2D)

dropout_2 (Dropout)          (None, 4, 4, 32)          0

flatten_6 (Flatten)          (None, 512)               0

dense_7 (Dense)              (None, 10)                5130

=================================================================
Total params: 53,034
Trainable params: 52,650
Non-trainable params: 384
```

Figure 5 - Final model architecture

3.5.Transfer learning

After reading some articles on transfer learning (V7labs.com, 2022), I decided to push myself to experiment with one more complexity level to improve my final model. Transfer learning relies on pre-trained models to speed optimization and serve as a starting point for new models.

Based on the Imagenet dataset (Image-net.org, 2017), which contains over 14 million labelled images, broken into 1000 classes for research purposes, various models are available to classify these images correctly. Some examples are Resnet, Inception and VGG. I decided to try the Resnet50 model from the Keras pre-trained models because of its speed, accuracy and lower number of parameters.

I added the Resnet50 to a Sequential model and rendered the Resnet50 layers untrainable (so my model would not retrain the Resnet50 weights and use the strength of the pre-trained weights). I then flattened the Resnet50 model and added various Dropout, Batch Normalisation and Dense layers before outputting a dense layer with a Softmax activation. Finally, I ran one model with 50 epochs with a batch size of 100 and recorded the results.

In all experiments, I used the Adam optimizer and categorical cross-entropy as my loss function since this is a multiclassification problem.

4. Results

The training and validation accuracy graphs for each model are shown in figure 6-14.

Figure 6 - Baseline model performance

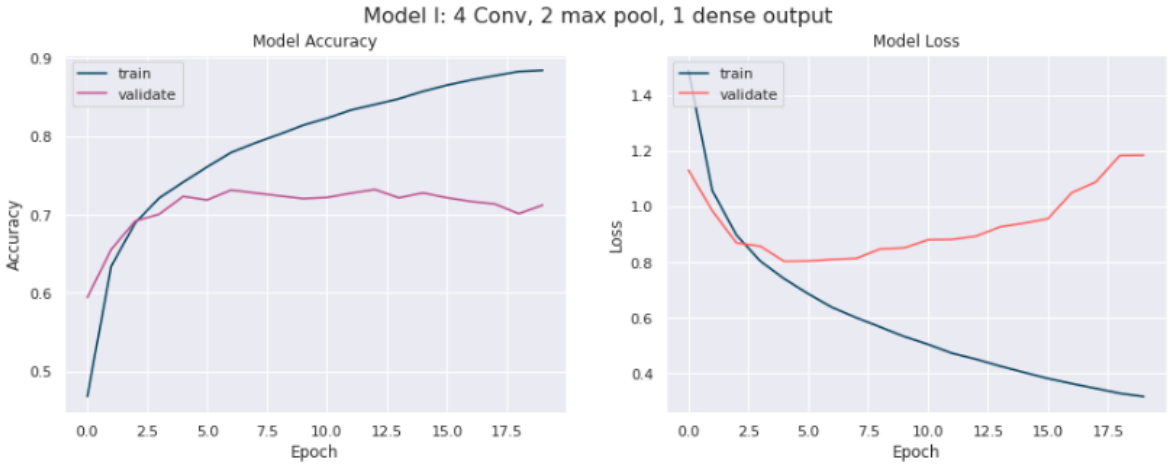

Figure 7 - Model 1 performance

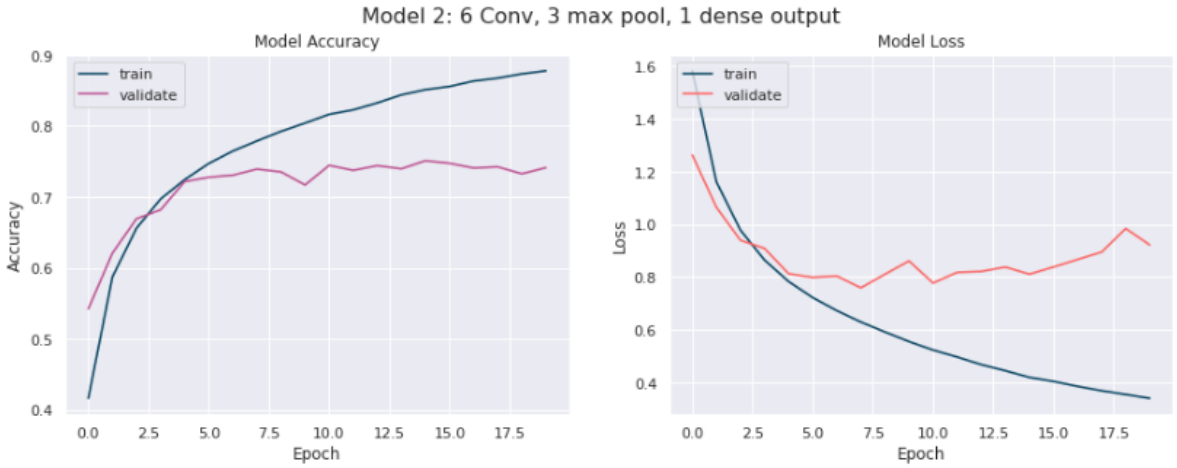

Figure 8 - Model 2 performance

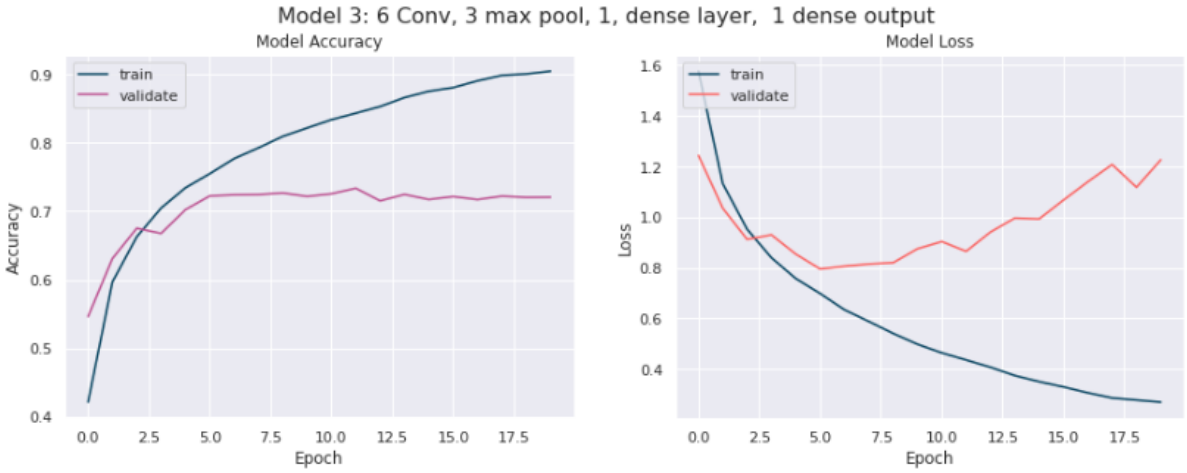

Figure 9 - Model 3 performance

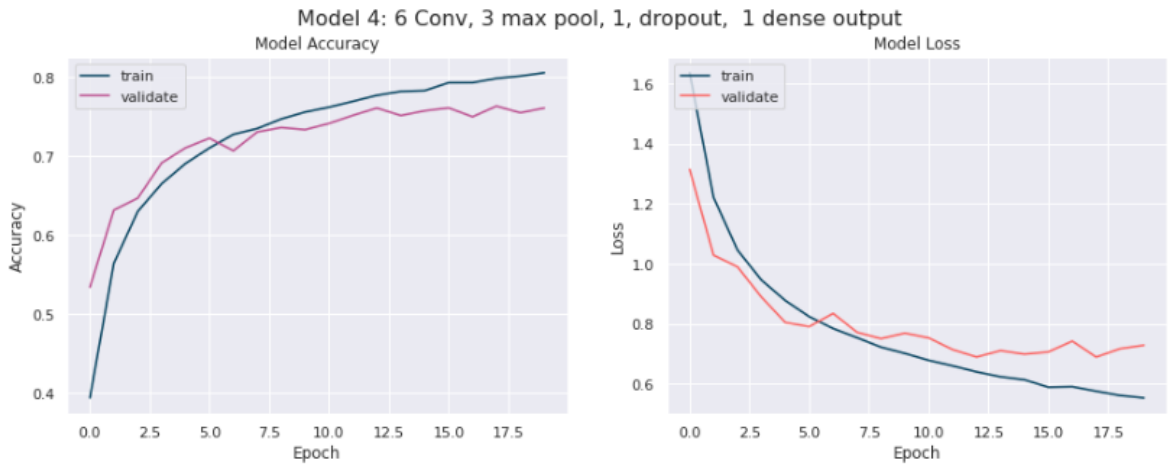

Figure 10 - Model 4 performance

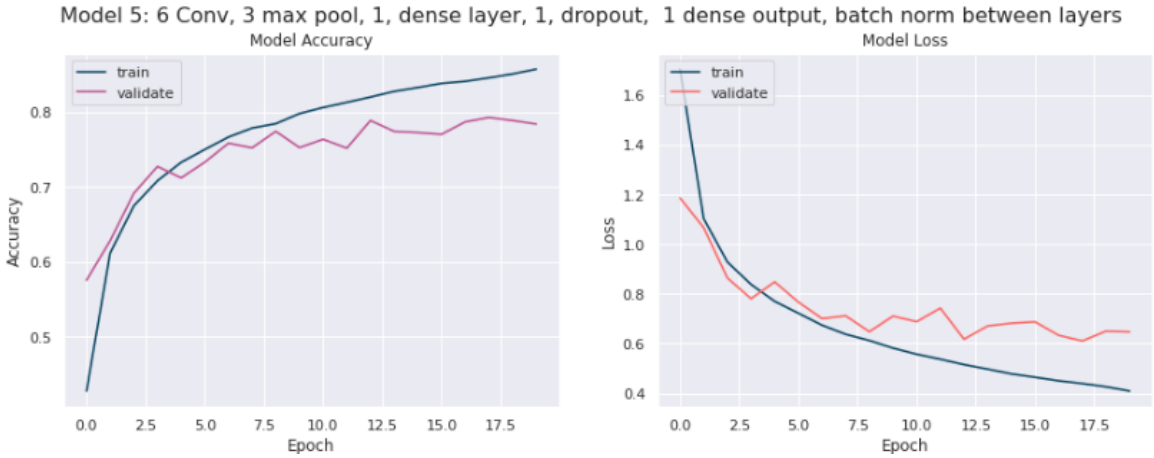

Figure 11 - Model 5 performance

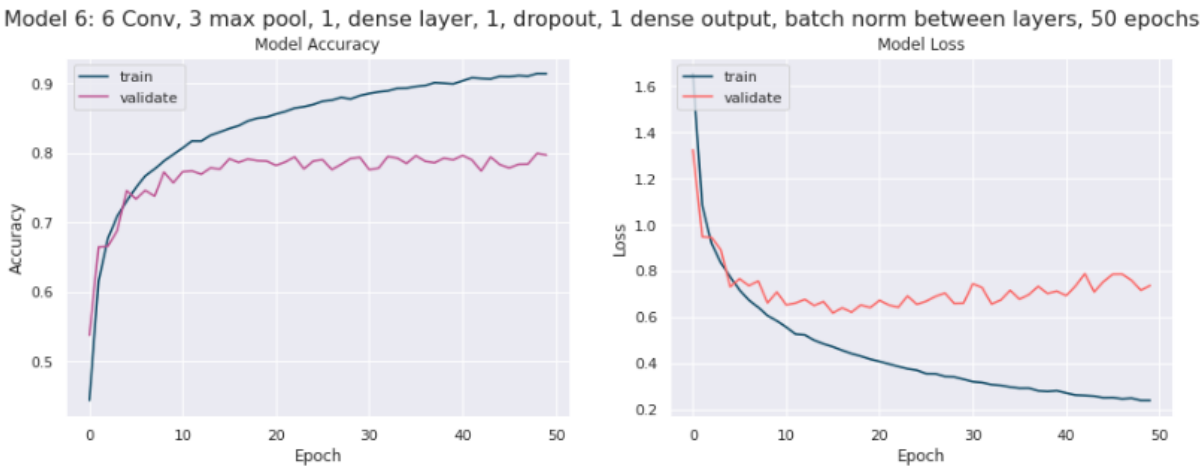

Figure 12 - Model 6 performance

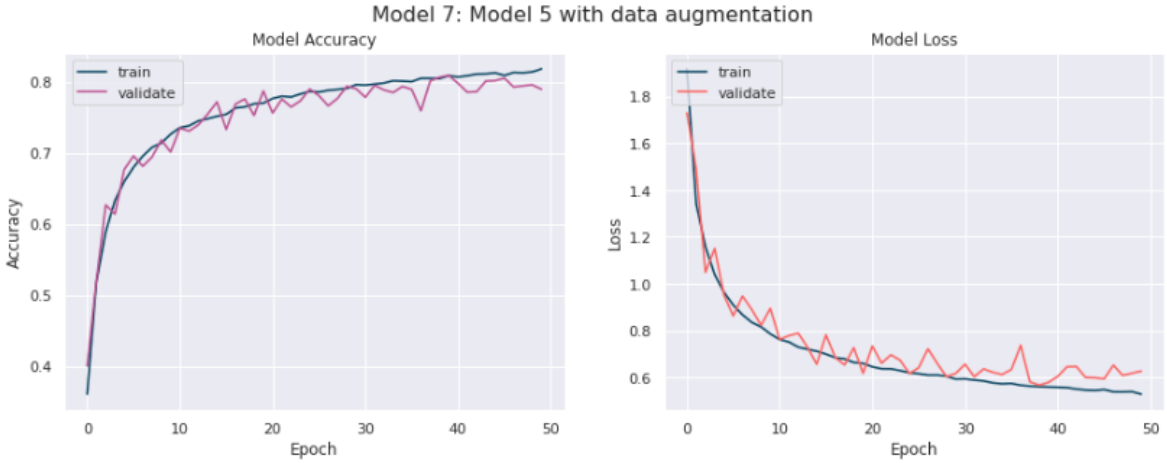

Figure 13 - Model 7 performance

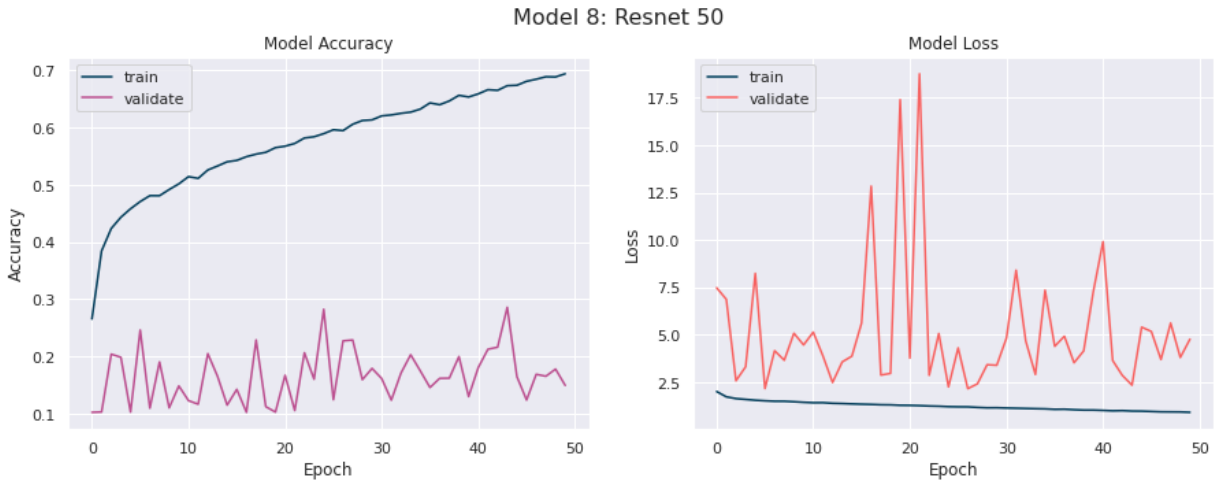

Figure 14 - Model 8 performance

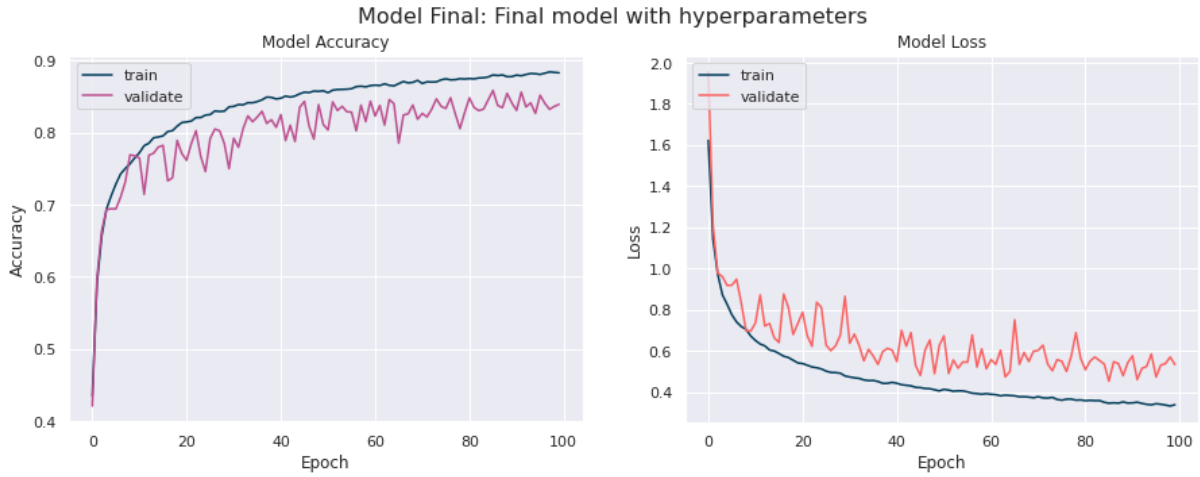

Figure 15 - Model 9 performance

A summary of the validation and training accuracy and loss scores are listed in Figure 16.

	Model	Train Accuracy	Validation Accuracy	Train Loss	Validation Loss
0	Baseline Model: 2 Conv, 1 max pool, 1 dense output	0.933525	0.6176	0.183276	2.158859
1	Model I: 4 Conv, 2 max pool, 1 dense output	0.888850	0.6985	0.309570	1.175378
2	Model 2: 6 Conv, 3 max pool, 1 dense output	0.877775	0.7398	0.339521	0.939233
3	Model 3: 6 Conv, 3 max pool, 1, dense layer, 1 dense output	0.905300	0.7225	0.259209	1.180293
4	Model 4: 6 Conv, 3 max pool, 1, dropout, 1 dense output	0.801825	0.7623	0.561029	0.711217
5	Model 5: 6 Conv, 3 max pool, 1, dense layer, 1, dropout, 1 dense output, batch norm between layers	0.855175	0.7795	0.413086	0.680963
6	Model 6: 6 Conv, 3 max pool, 1, dense layer, 1, dropout, 1 dense output, batch norm between layers, 50 epochs	0.914900	0.7898	0.234914	0.791662
7	Model 7: Model 5 with data augmentation	0.819450	0.8136	0.521565	0.544821

Figure 16 - Summary of all model results

5. Evaluation

My baseline model gave a training accuracy of 94 % vs a validation accuracy of 62 %, clearly indicating that the training set is fitting well. Still, it was not generalising very well to unseen data (Figure 6)

The addition of 2 more Conv2D layers and a Max Pool layer increased the validation accuracy to 70 %, and the training accuracy decreased to 89. In addition, the extra Conv2D layers allowed the model to extract more low-level features of the images, thus increasing the performance on new data (Figure 7)

Adding another block of 2 Conv2D layers and 1 Max Pool layer again increased the validation accuracy. However, this time it went up to 74 %, while the training accuracy stayed reasonably constant at 88 % (Figure 8).

I only added a Dense layer in the following experiment. Training accuracy increased to 91 %, but validation accuracy dropped to 72. The drop is a clear indication that overfitting is happening again (Figure 9).

I felt at this stage that the model might benefit from some regularisation. "Regularisation imposes a penalty on model's complexity or smoothness, allowing for good generalisation to unseen data even when training on a finite training set or with an inadequate iteration." (Tian and Zhang, 2022). So I added a Dropout layer with a value of 0.4 just before the Flatten layer. I was very impressed with this result: the validation accuracy increased to 76 %, and this was at the

expense of the training accuracy dropping to 80 %. Keeping these two metrics close together is a good sign (Figure 10).

In the following experiment, I kept the Dropout layer and added Batch Normalization after each Conv2D layer. Batch Normalisation adjusts the mean and scale of the activations to equalise the input layer and hidden layers. Deep neural networks may employ a faster learning rate without disappearing or exploding gradients because of this levelling effect with the additional layers. As a result, validation accuracy went up to 79 %, and training accuracy went up to 85 % (Figure 11).

For a final test I increased the number of epochs from 20 to 50, to give the model more time to train. The validation accuracy increased to 79% and the training accuracy to 91%. More training time has increased the model's value, but has also increased the training time drastically (Figure 12).

Due to time and computational constraints, I decided to keep this as my final architecture and move on to other methods to improve this model.

Data augmentation approached the problem of overfitting by focusing on the core training dataset by enlarging it (Shorten and Khoshgoftaar, 2019). This process allows the model to learn the intricacies of the features and results in an ability to generalise well on unseen data. Next, I ran the Keras ImageDataGenerator and the Flow() function to generate the additional images. An example of the result of the augmentation, before and after is shown in Figure 17.

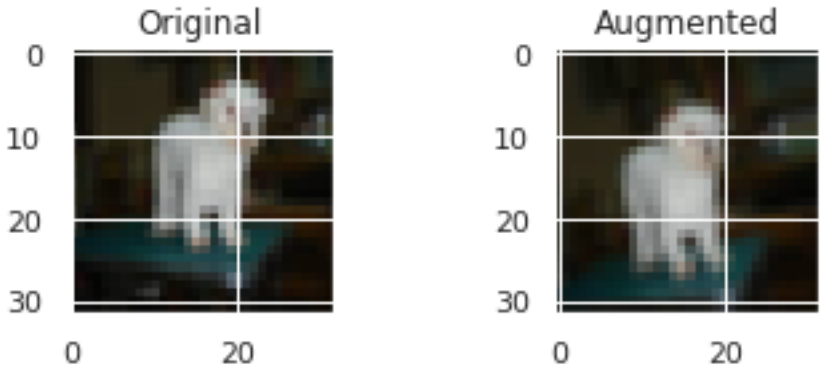

Figure 17 - Augemented image before and after

Taking the final architecture and adding the augmented images resulted in an increase in validation accuracy to 81 % and training accuracy of 82 %. The additional images increased the model's ability to predict unseen data (Figure 13).

The last step was to tune some of the hyperparameters to improve and stabilise the validation accuracy. The result of testing optimal values for the number of filters, dropout rate, learning rate and activation function is shown in Figure 18.

```
{'filters1': 64, 'filters2': 48, 'filters3': 48, 'filters4': 48, 'filters5': 48, 'filters6':
48, 'dropout': 0.0, 'activation': 'relu', 'learning_rate': 0.01}
```

Figure 18 - Keras Tuner results

After entering these hyperparameters into the final augmented model, the validation accuracy increased to 84 % and training accuracy to 88 % (Figure 15)

I then took this model and evaluated it on the hold-out test set, and achieved a testing accuracy of 84 %. Figure 19 shows sample predictions using this model, any incorrect predictions are highlighted in red (in this case there are no incorrect predictions!)

Figure 19 - Predictions on the test dataset

My extra experiment using transfer data on the unaugmented data did not prove as successful. Validation accuracy was extremely low at 15 % and the training accuracy at 62 %. I suspect this is due to my architecture for this model not working very well and something to experiment with in more detail in the future (Figure 14).

A note on computation speeds:

As my model got more complicated, my training time increased drastically. Working with a graphics processing unit (GPU) on Kaggle and Google Colab helped immensely, but some models' training times took hours and hours to complete. For example, the hyperparameter tuning took over 1 hour to complete each time I ran it. The baseline model took 146 seconds to run, while the final augmented model with hyperparameters took 1898 seconds.

6. Ethical considerations in computer vision

Facial recognition software is one of the most powerful surveillance tools ever created. It maps, analyses, and then confirms the identity of a face in a photograph or video. While many people use facial recognition to unlock their phones or sort their photos, how businesses and governments use it has a far more significant impact on people's lives.

While you may be able to opt-out of or turn off facial recognition on a device you own or software you use, the ubiquity of cameras makes the technology increasingly challenging to avoid in public. Concerns about how widely facial recognition software is used, exacerbated by evidence of racial profiling and protester identification, have prompted major corporations such as Amazon, I.B.M., and Microsoft to halt sales of their software to law enforcement. (Klosowski, 2020). However, as restrictions expire and facial recognition technology improves and becomes more affordable, society will be forced to answer big questions about how facial recognition should be regulated and small questions about which services we're willing to use and which privacy sacrifices we're ready to make.

This unrestrained spread of A.I. applications into our lives can feel invasive. For example, facial recognition and computer vision use personal data to improve our lives in unexpected ways. However, how personal information is used raises privacy, discrimination, and security concerns. In an ongoing effort to harness the power of computer vision with minimal collateral damage, we examine some of these issues and discuss solutions.

Computer vision uses images to allow artificial intelligence systems to recognise objects, faces and movements to make inferences or classification decisions. However, despite the usefulness of these systems, there is a range of ethical concerns lingering in this technology's future. The variety of photos fed into the system significantly impacts the system's accuracy during the analysis and recognition steps.

For example, suppose the sample sets consist primarily of white men, as was the case in the early training of facial recognition systems. In that case, the programs will struggle to identify BIPOC (Black, Indigenous, and People of Colour) faces and women accurately. In recent years, the best facial recognition software has begun to correct this, but white males are still falsely matched less frequently than others. Asian and Black men are more likely to be misidentified than white men.

Mutale Nkonde, a Stanford Digital Civil Society Lab fellow and member of the TikTok Content Advisory Council, observes that even if the systems are flawless, issues with gender identification persist: "Most labels are binary: male or female. There is no way for that system to consider non-binary people or people who have transitioned." (Klosowski, 2020)

After a company trains its software to detect and recognise faces, it can locate and compare those faces to those in a database. In this identification step, the software searches a database of photos and cross-references to identify a person based on pictures from various sources, such as mug shots and images scraped from social media. It then displays the results, which are usually ranked by accuracy. These systems appear complicated, but you can build your facial recognition system using off-the-shelf software with some technical knowledge.

The modern era of facial recognition began in the 2010s when computers became powerful enough to train the neural networks needed to recognise faces. In 2011, facial recognition confirmed Osama bin Laden's identity. (U.S. military deploys facial recognition technology in Bin Laden operation, 2011). In 2014, Facebook made public its DeepFace photo-tagging software (Chowdhry, 2014), the same year that Edward Snowden released documents revealing the extent to which the U.S. government was collecting images to build a database. (Risen and Poitras, 2014)

In 2015, Windows Hello introduced facial recognition as a security feature for personal devices (Pullen, 2015), followed by Android's Trusted Face (later removed due to security concerns) (El Khoury, 2019). Apple's Face ID was launched with the iPhone X in 2017. (apple.com, n.d)

Because computer vision can identify faces, objects, locations, and movements, it raises ethical and privacy concerns, including fraud, bias, inaccuracy, and uninformed consent. For example:

Hackers and fraudsters have used masks and photos to fool facial recognition technology and claim benefits or gain access to sites in another person's name.

Facial recognition has an inherent bias, as it produces far more false identifications of Black and Asian faces than white faces, increasing the risk of false arrest. It also identifies elderly and young children more than middle-aged adults, skewing evidence and investigations.

Extraneous signals or data noise can cause inaccuracy in the models resulting in incorrect outputs and conclusions being drawn and reported.

There are also ethical concerns over consent violations of privacy laws. For example, facial recognition collects personal data without consent in some countries, resulting in numerous class action lawsuits. The controversial CV-driven software designed to detect and report Child Sexual Abuse Material (CSAM) found on personal devices has been delayed by Apple. Users feared that their images would be used for government surveillance or wrongful arrest. (Wakefield, 2021)

Researchers collect large data sets of facial images without consent, often used to improve military and commercial surveillance algorithms.

Surveillance applications worldwide, including China, who started using personal data posted on the internet as training data, which has since evolved into a commercial credit system aimed at social engineering behaviour. Chinese citizens are assigned a number: good deeds earn points, while bad deeds cost them, with perks and drawbacks.

Even if your offence is non-financial, such as a neighbour dispute, a pilot system prevents you from obtaining a loan or mortgage. Giving blood, for example, can lower the interest rate on your loan. (Campbell, 2019)

In China, social credit scoring is best understood as an ideology that includes both punishments and rewards to improve governance and eliminate disorder and fraud. Commercial schemes deal with perks, whereas state schemes deal with punishment. Both organisations collaborate to promote social responsibility.

They are, however, intrusive.

Government agencies collect and disseminate judgment data. Instead of adding a new layer of arbitrary regulation, the system uses technology to enforce existing legal prohibitions through shame and hardship.

It is also a question of what point it benefits the Chinese government eager to suppress dissent and silence critics. As the state continues to persecute religious minorities and academics (Campbell, 2019), one wonders what social-credit systems will be next on the list.

7. Conclusion

The Cifar-10 dataset is a benchmark dataset for computer vision learning and experimentation; the current best test accuracy score is shown in the CaiT-M-36 U 224 paper with a score of 99.4 % (Touvron et al., n.d.), indicating that my best model of 84 % has a lot of room for improvement.

This group of experiments showed the value of regularisation, hyperparameter tuning and architecture construction. It also highlighted the importance of considerations regarding computational power in running experiments. I also learned that I should have been saving the weights of my models after each iteration to save training time for consecutive experiments (possibly using Pickle or Kera's 'save' function).

In future experiments, I would like to try putting more into transfer learning and building a model that harnesses some of the popular pre-made results. Transformers are used predominantly in natural language processing (NLP); an application of these called Vision Transformers has been shown to attain excellent results compared to CNN's (Dosovitskiy et al., n.d.). It would be interesting to see how these transformers fare in comparison or in addition to my current model architecture.

Computer vision is already used extensively in facial recognition, anomaly detection and sensor data and has a bright future in areas such as AI explainability and workplace safety.

Therefore, learning how to optimise these models and decrease computational needs will focus on the machine learning community in the near future.

Appendix 1

Libraries and packages (with their import code) used in this assignment.

```
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
import numpy as np
import pandas as pd
import time
from tensorflow.keras import datasets,models,layersfrom tensorflow.keras.preprocessing.image
import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten, BatchNormalization,
Conv2D, MaxPooling2D
from tensorflow.keras.layers import SpatialDropout2D
from tensorflow.keras import layers
from tensorflow import keras
from tensorflow.keras import regularizers
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import keras_tuner as kt
from tensorflow.keras.optimizers import SGD, Adam
from tensorflow.keras.wrappers.scikit_learn import KerasClassifier
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
```

import seaborn as sns

import matplotlib

import matplotlib.pyplot as plt

References

Krizhevsky, A. and Hinton, G. (2009a) CIFAR-10 and CIFAR-100 Dataset, Toronto.edu. Available at: https://www.cs.toronto.edu/~kriz/cifar.html.

Krizhevsky, A. and Hinton, G. (2009b) "CIFAR-10 Python Version," Learning Multiple Layers of Features from Tiny Images. Available at: https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz.

Baheti, P. (2022). *A Newbie-Friendly Guide to Transfer Learning: Everything You Need to Know*. [online] V7labs.com. Available at: https://www.v7labs.com/blog/transfer-learning-guide#:~:text=let's%20dive%20in!-,What%20is%20Transfer%20Learning%3F,when%20modeling%20the%20second%20task. [Accessed 24 Apr. 2022].

Data augmentation (2022). *Data augmentation | TensorFlow Core*. [online] TensorFlow. Available at: https://www.tensorflow.org/tutorials/images/data_augmentation [Accessed 30 Apr. 2022].

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J. and Houlsby, N. (n.d.). *AN IMAGE IS WORTH 16X16 WORDS: TRANSFORMERS FOR IMAGE RECOGNITION AT SCALE*. [online] Available at: https://arxiv.org/pdf/2010.11929.pdf [Accessed 30 Apr. 2022].

https://www.facebook.com/jason.brownlee.39 (2019). *Accelerate the Training of Deep Neural Networks with Batch Normalization*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/ [Accessed 30 Apr. 2022].

https://www.facebook.com/MachineLearningMastery (2019). *Understand the Impact of Learning Rate on Neural Network Performance*. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/ [Accessed 30 Apr. 2022].

Image-net.org. (2017). *ImageNet*. [online] Available at: https://image-net.org/challenges/LSVRC/index.php [Accessed 24 Apr. 2022].

kutaykutlu (2021). *ResNet50 Transfer Learning CIFAR-10 Beginner*. [online] Kaggle.com. Available at: https://www.kaggle.com/code/kutaykutlu/resnet50-transfer-learning-cifar-10-beginner [Accessed 26 Apr. 2022].

Nair, A. (2022). *Baseline Models: Your Guide For Model Building - Towards Data Science*. [online] Medium. Available at: https://towardsdatascience.com/baseline-models-your-guide-for-model-building-1ec3aa244b8d#:~:text=A%20baseline%20model%20is%20essentially,may%20have%20little%20predictive%20power. [Accessed 24 Apr. 2022].

Nikhil B (2017). *Image Data Pre-Processing for Neural Networks - Becoming Human: Artificial Intelligence Magazine*. [online] Medium. Available at: https://becominghuman.ai/image-data-pre-processing-for-neural-networks-498289068258 [Accessed 7 Apr. 2022].

Olugbenga, M. (2021). *Balanced Accuracy: When Should You Use It? - neptune.ai*. [online] neptune.ai. Available at: https://neptune.ai/blog/balanced-accuracy [Accessed 24 Apr. 2022].

Shorten, C. and Khoshgoftaar, T.M. (2019). A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, [online] 6(1). Available at: https://link.springer.com/article/10.1186/s40537-019-0197-0?code=a6ae644c-3bfc-43d9-b292-82d77d5890d5 [Accessed 30 Apr. 2022].

the (2022). *Machine Learning for Developers*. [online] O'Reilly Online Learning. Available at: https://www.oreilly.com/library/view/machine-learning-for/9781786469878/252b7560-e262-49c4-9c8f-5b78d2eec420.xhtml [Accessed 30 Apr. 2022].

Thompson, N., Greenewald, K., Lee, K. and Manso, G. (n.d.). *The Computational Limits of Deep Learning*. [online] Available at: https://arxiv.org/pdf/2007.05558.pdf.

Tian, Y. and Zhang, Y. (2022). A comprehensive survey on regularization strategies in machine learning. *Information Fusion*, [online] 80, pp.146–166. Available at: https://www.sciencedirect.com/science/article/abs/pii/S156625352100230X [Accessed 30 Apr. 2022].

Touvron, H., Cord, M., Synnaeve, G., Jégou, H. and Ai, F. (n.d.). *Going deeper with Image Transformers*. [online] Available at: https://arxiv.org/pdf/2103.17239v2.pdf [Accessed 30 Apr. 2022].

Urban, C. and Miné, A. (2021). A Review of Formal Methods applied to Machine Learning. *arXiv.org*. [online] Available at: https://arxiv.org/abs/2104.02466 [Accessed 24 Apr. 2022].

Klosowski, T. (2020). *Facial recognition is everywhere. Here'sHere's what we can do about it.* [online] Wirecutter: Reviews for the Real World. Available at: https://www.nytimes.com/wirecutter/blog/how-facial-recognition-works/ [Accessed 24 Apr. 2022].

U.S. military deploys facial recognition technology in Bin Laden operation. (2011). *Biometric Technology Today*, [online] 2011(5), p.1. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0969476511700819 [Accessed 21 Feb. 2020].

Chowdhry, A. (2014). *Facebook'sFacebook's DeepFace Software Can Match Faces With 97.25% Accuracy*. [online] Forbes. Available at: https://www.forbes.com/sites/amitchowdhry/2014/03/18/facebooks-deepface-software-can-match-faces-with-97-25-accuracy/?sh=6348971c54fc [Accessed 24 Apr. 2022].

Risen, J. and Poitras, L. (2014). N.S.A. Collecting Millions of Faces From Web Images. *The New York Times*. [online] 31 May. Available at: https://www.nytimes.com/2014/06/01/us/nsa-collecting-millions-of-faces-from-web-images.html.

Pullen, J.P. (2015). *How Windows 10 Could Kill Passwords Forever*. [online] Time. Available at: https://time.com/4128834/windows-10-hello-facial-recognition/ [Accessed 24 Apr. 2022].

El Khoury, R. (2019). *Trusted Face smart unlock method has been removed from Android devices*. [online] Android Police. Available at: https://www.androidpolice.com/2019/09/04/trusted-face-smart-unlock-method-has-been-removed-from-android-devices/#:~:text=Android%20Smart%20Lock [Accessed 24 Apr. 2022].

Wakefield, J. (2021). Apple delays plan to scan iPhones for child abuse. *B.B.C. News*. [online] 3 Sep. Available at: https://www.bbc.com/news/technology-58433647 [Accessed 24 Apr. 2022].

Campbell, C. (2019). *How China Is Using Big Data to Create a Social Credit Score*. [online] Time.com. Available at: https://time.com/collection/davos-2019/5502592/china-social-credit-score/

Albawi, S., Mohammed, T.A. and Al-Zawi, S. (2017). Understanding of a convolutional neural network. *2017 International Conference on Engineering and Technology (ICET)*. [online] Available at: https://ieeexplore.ieee.org/abstract/document/8308186 [Accessed 30 Apr. 2022].