CIFAR-10 image recognition task

The CIFAR-10 data

Data splits

The data (n=60,000) was split into a training set (n=50,000) and a validation set (n=10,000)

Rationale for validation

Neural networks are prone to over-fit the training data (memorise)

Performance of the model must be assessed in validation data that the network has not seen during training

Data structure

Shape of each data instance was (32, 32, 3)

32 x 32 pixels, three channels (RGB)

The values were normalised by division by 255 (to range 0-1)

10 classes (labels)

Approach to model building experiments

ANN → CNN → Deeper CNN

Initial test

Test the performance of a neural network without any convolutional layers

This will give a baseline performance for comparison

Add convolution layers

Test the change in performance with a small number of convolutional layers

Visualise the kernels and feature maps learned

Deep CNN with many layers including dropout, batch normalization

Optimise for performance on the classification task

Python keras package used for model building

Modelling choices

Loss function

Sparse categorical cross entropy loss

This is appropriate for a multi-class classification objective

Metrics

Accuracy

This is appropriate for a balanced multi-class problem

NB if classes were imbalance this would not be appropriate

Optimizer

Stochastic gradient descent

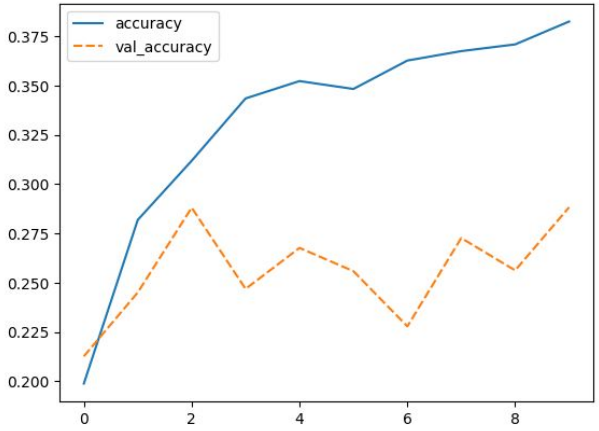
Ten epochs

Chosen empirically as in initial models, accuracy plateaus before ten epochs

ANN: from one to three FC layers

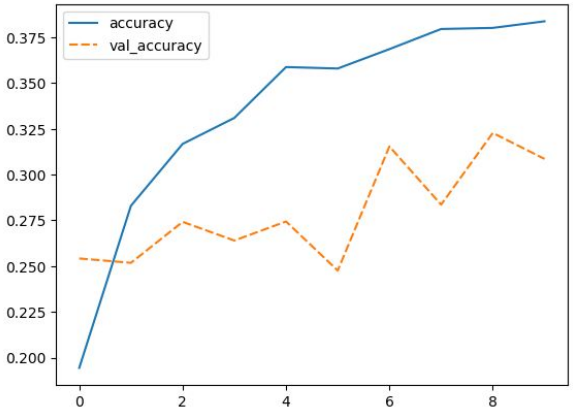
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 32, 32, 3)]	0
flatten (Flatten)	(None, 3072)	0
dense (Dense)	(None, 10)	30730

Total params: 30,730
Trainable params: 30,730
Non-trainable params: 0



Layer (type)	Output Shape	Param #
input_3 (InputLayer)	[(None, 32, 32, 3)]	0
flatten_2 (Flatten)	(None, 3072)	0
dense_3 (Dense)	(None, 100)	307300
dense_4 (Dense)	(None, 50)	5050
dense_5 (Dense)	(None, 10)	510

Total params: 312,860
Trainable params: 312,860
Non-trainable params: 0

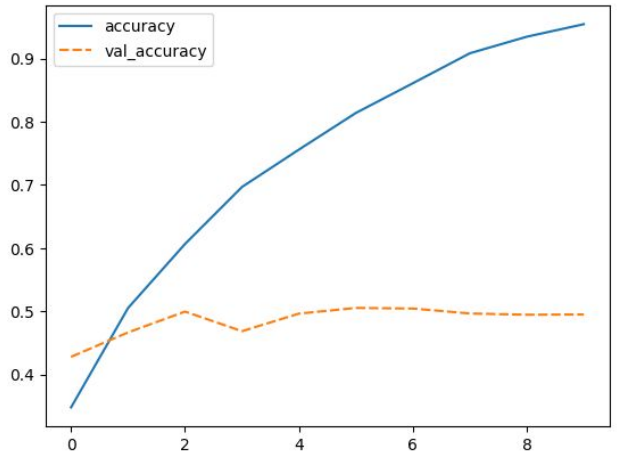


CNN: one convolutional layer

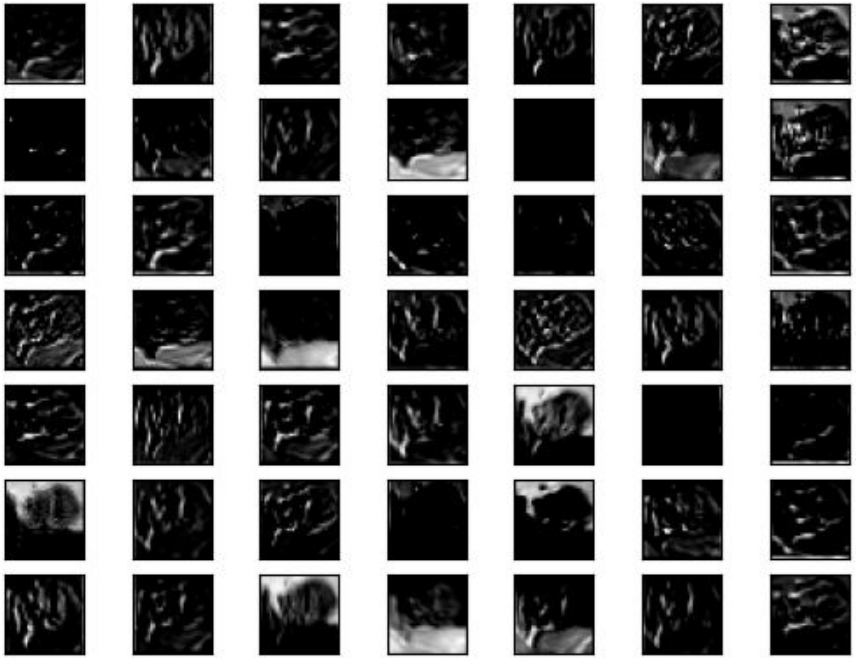
- 1 convolutional layer
- 50 kernels
- Kernel size = (3,3)
- Stride = 1
- Activation = ReLU
- Optimizer = Adam (Kingma et al. 2015)
- Metric = accuracy
- 1 FC layer for classification

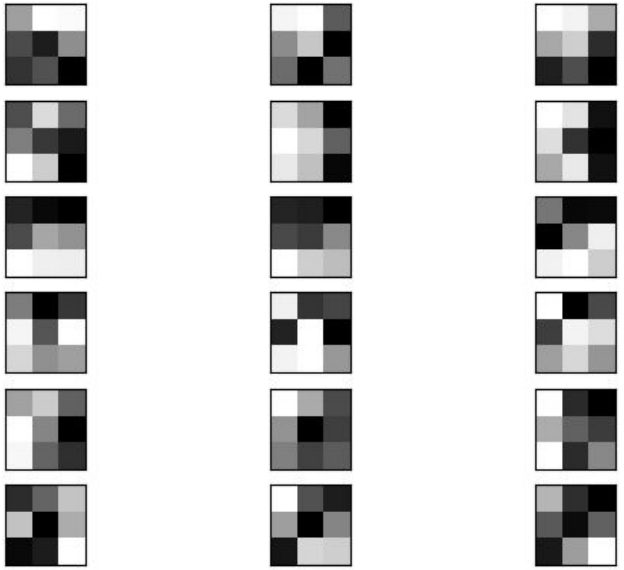
Layer (type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 32, 32, 3)]	0
conv2d (Conv2D)	(None, 32, 32, 50)	1400
flatten_3 (Flatten)	(None, 51200)	0
dense_6 (Dense)	(None, 10)	512010

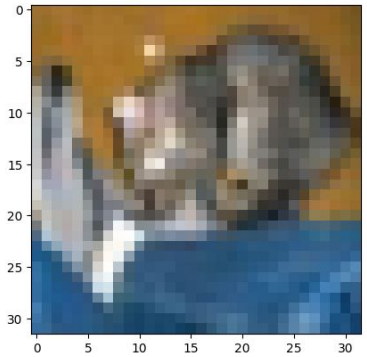
Total params: 513,410
Trainable params: 513,410
Non-trainable params: 0



CNN: one convolutional layer







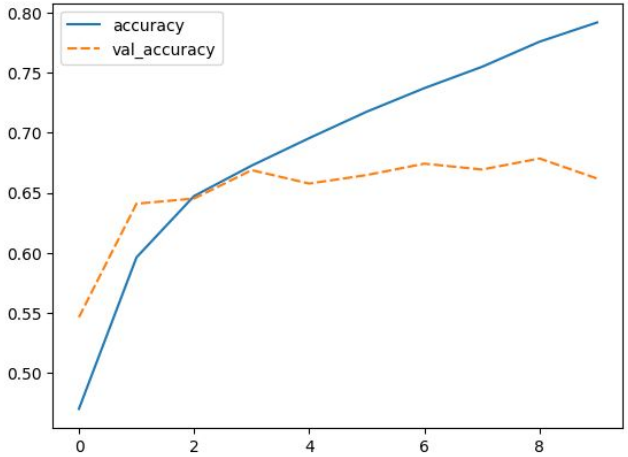
cat

CNN: 3 convolutional layers

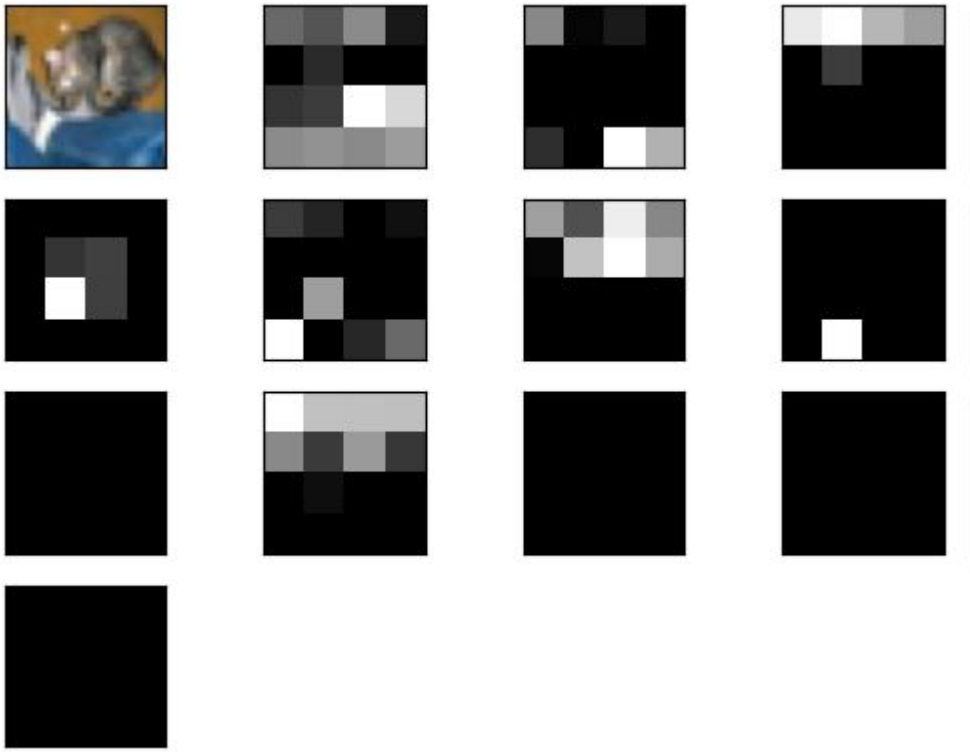
- 3 convolutional layers
- 50, 100, 200 kernels
- Kernel (3,3) stride 1
- Max pooling layers size (2,2)
- 3 FC layers: 40, 40, 10
- Activation = ReLU
- Optimizer = Adam
- Metric = accuracy

Layer (type)	Output Shape	Param #
input_15 (InputLayer)	[(None, 32, 32, 3)]	0
conv2d_40 (Conv2D)	(None, 32, 32, 50)	1400
max_pooling2d_39 (MaxPooling2D)	(None, 16, 16, 50)	0
conv2d_41 (Conv2D)	(None, 16, 16, 100)	45100
max_pooling2d_25 (MaxPooling2D)	(None, 8, 8, 100)	0
conv2d_42 (Conv2D)	(None, 8, 8, 200)	180200
max_pooling2d_26 (MaxPooling2D)	(None, 4, 4, 200)	0
flatten_14 (Flatten)	(None, 3200)	0
dense_32 (Dense)	(None, 40)	128040
dense_33 (Dense)	(None, 40)	1640
dense_34 (Dense)	(None, 10)	410

Total params: 356,790
Trainable params: 356,790
Non-trainable params: 0



CNN: 3 convolutional layers



1st convolutional layer

Max pooling

2nd convolutional layer

Max pooling

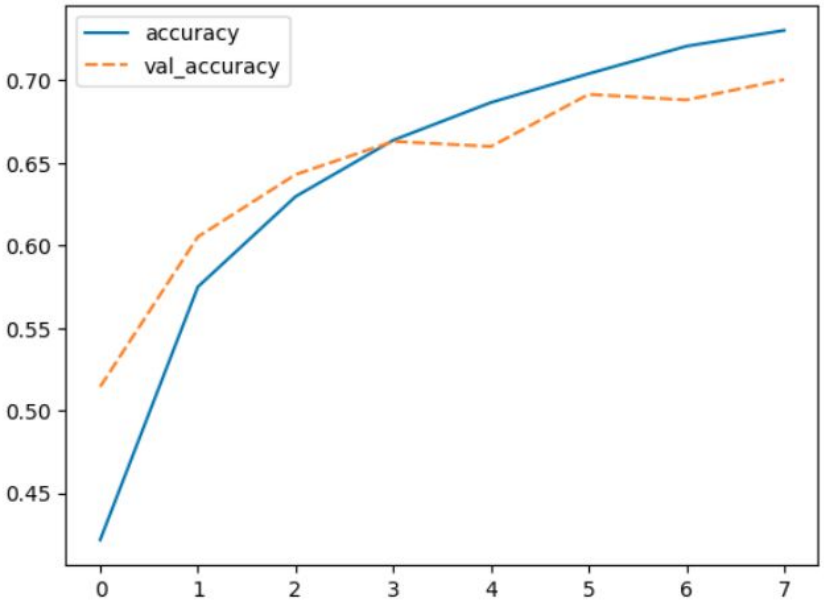
3rd convolutional layer

Max pooling

CNN: adding dropout

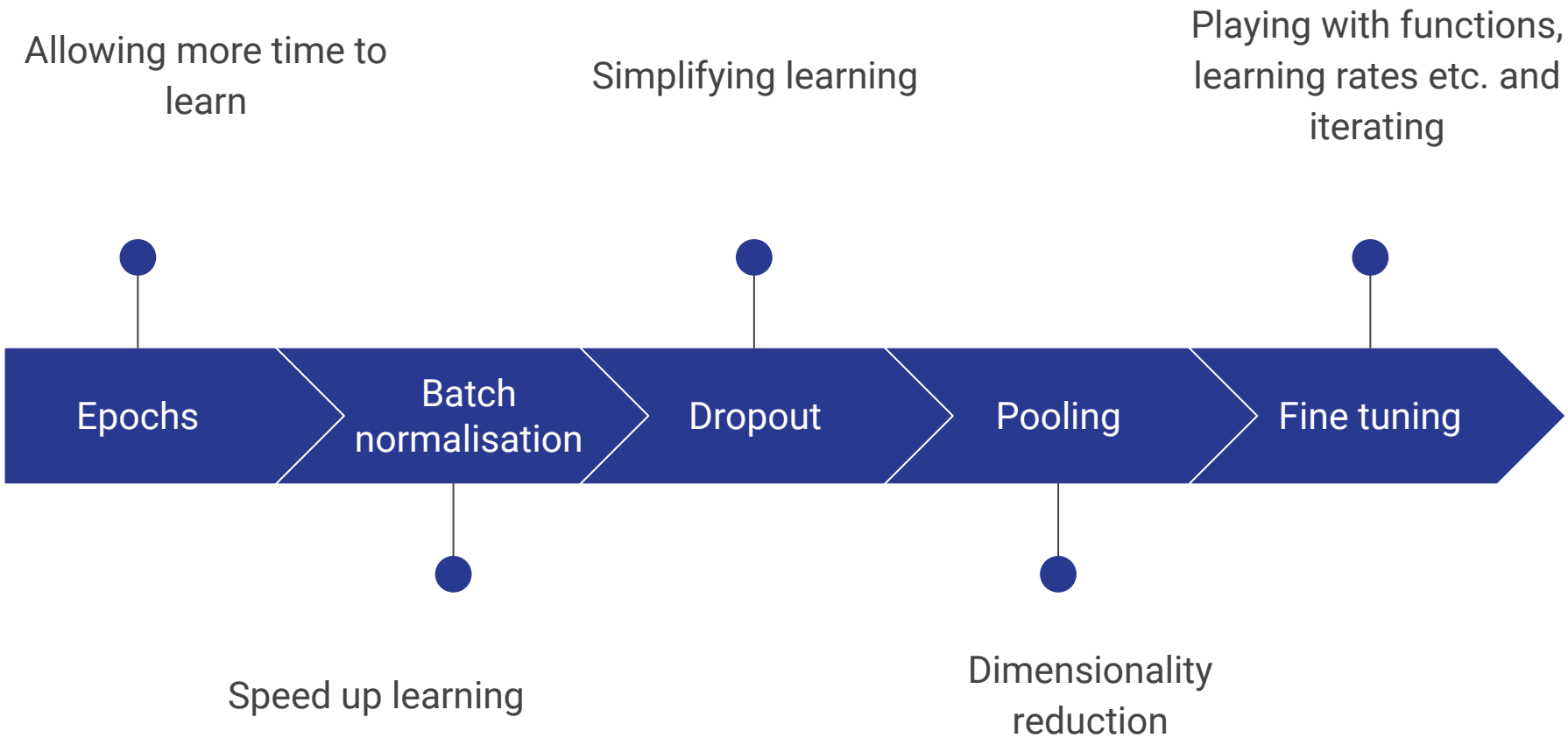
Layer (type)	Output Shape	Param #
input_37 (InputLayer)	[(None, 32, 32, 3)]	0
conv2d_144 (Conv2D)	(None, 32, 32, 50)	1400
max_pooling2d_88 (MaxPooling2D)	(None, 16, 16, 50)	0
conv2d_145 (Conv2D)	(None, 16, 16, 100)	45100
max_pooling2d_89 (MaxPooling2D)	(None, 8, 8, 100)	0
conv2d_146 (Conv2D)	(None, 8, 8, 200)	180200
conv2d_147 (Conv2D)	(None, 8, 8, 200)	360200
conv2d_148 (Conv2D)	(None, 8, 8, 100)	180100
max_pooling2d_90 (MaxPooling2D)	(None, 4, 4, 100)	0
flatten_36 (Flatten)	(None, 1600)	0
dropout_1 (Dropout)	(None, 1600)	0
dense_98 (Dense)	(None, 40)	64040
dense_99 (Dense)	(None, 40)	1640
dense_100 (Dense)	(None, 10)	410

Total params: 833,090
Trainable params: 833,090
Non-trainable params: 0



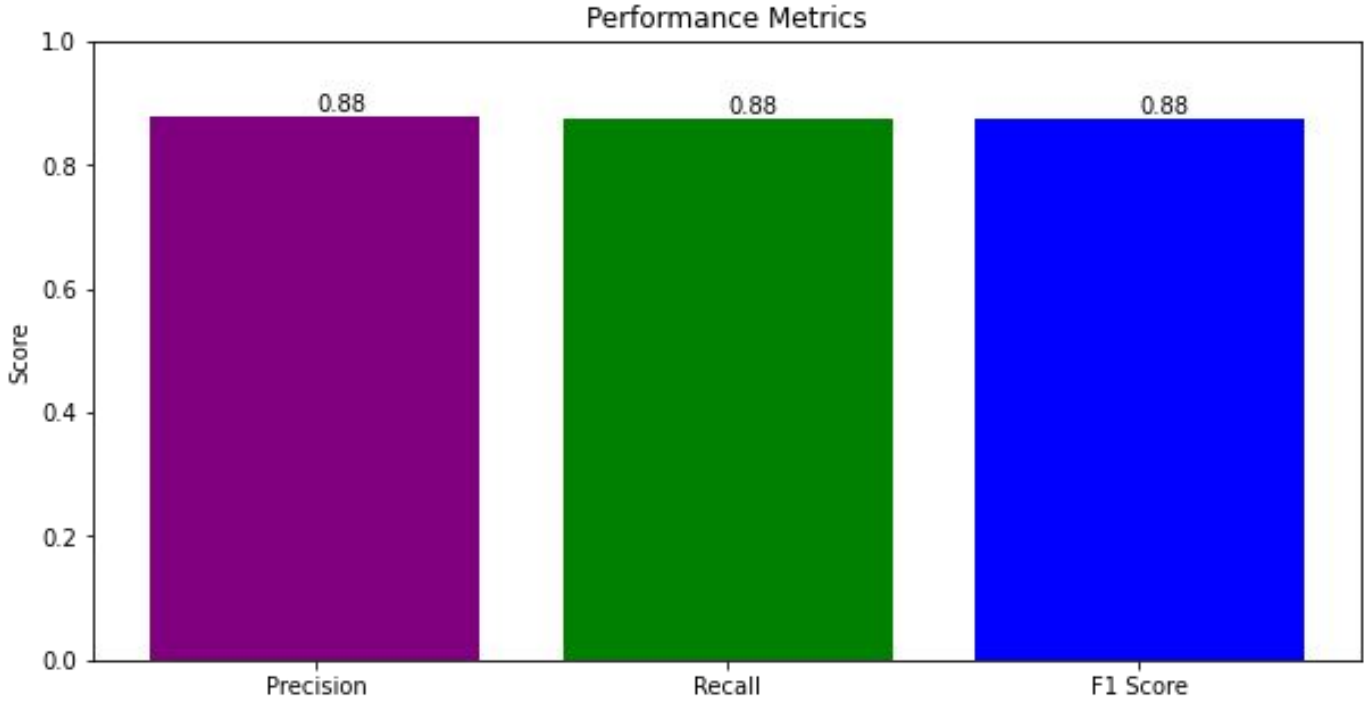
CNN PERFORMANCE OPTIMISATION

Optimising for performance

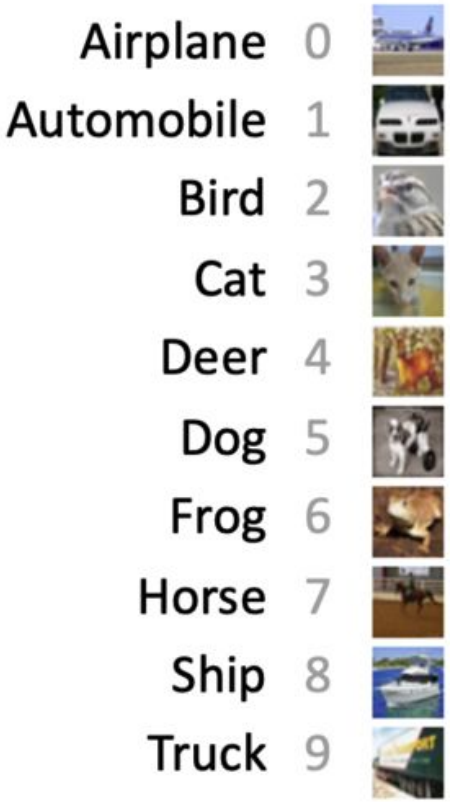


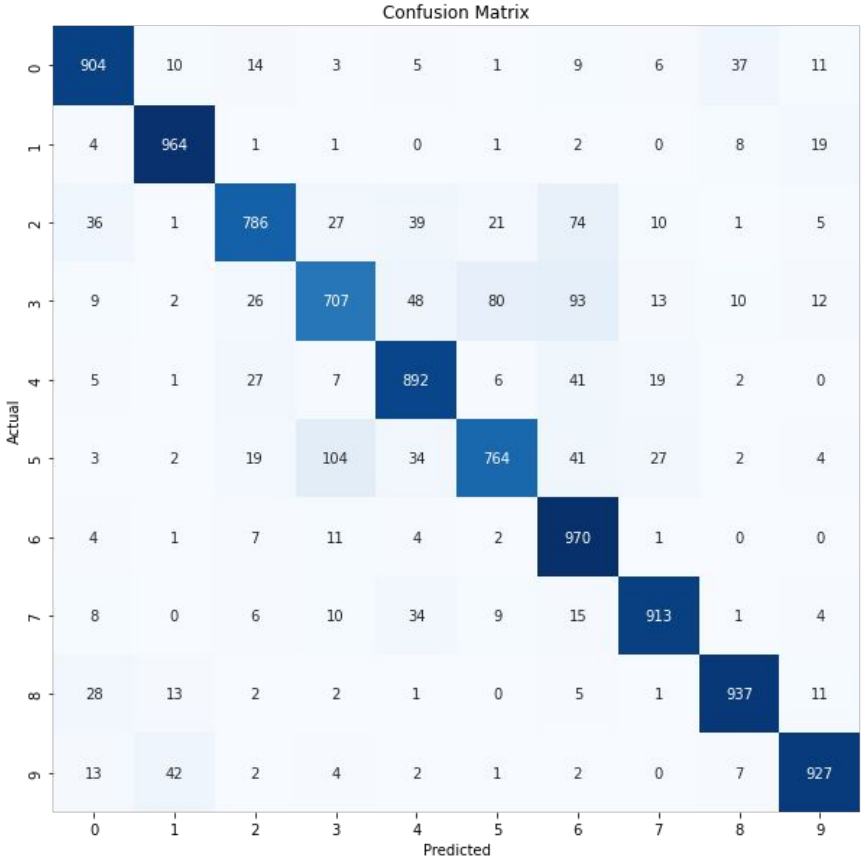
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (BatchN  (None, 32, 32, 32)       128
 ormalization)

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 batch_normalization_1 (Batc  (None, 32, 32, 32)       128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 dropout (Dropout)           (None, 16, 16, 32)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496
```

Performance metrics

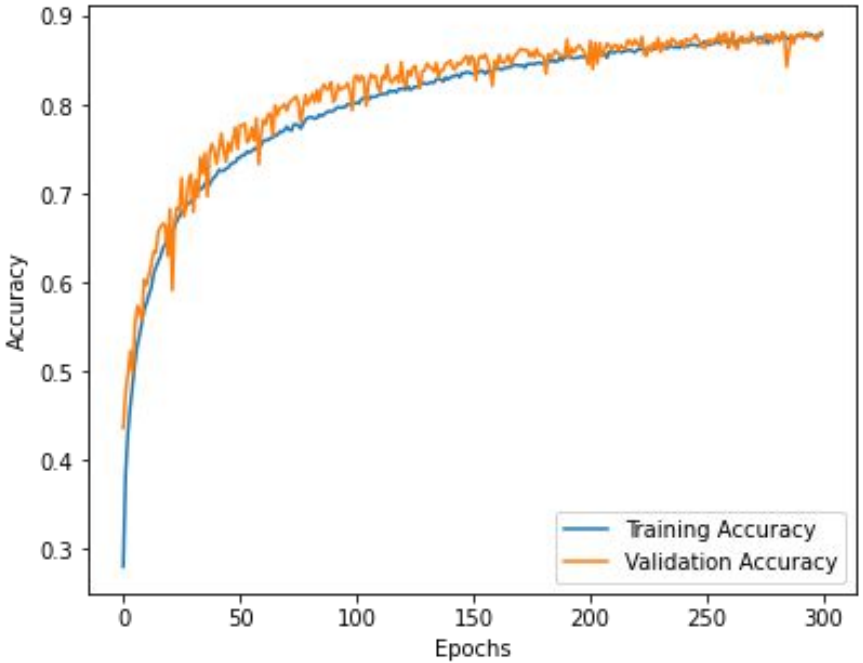


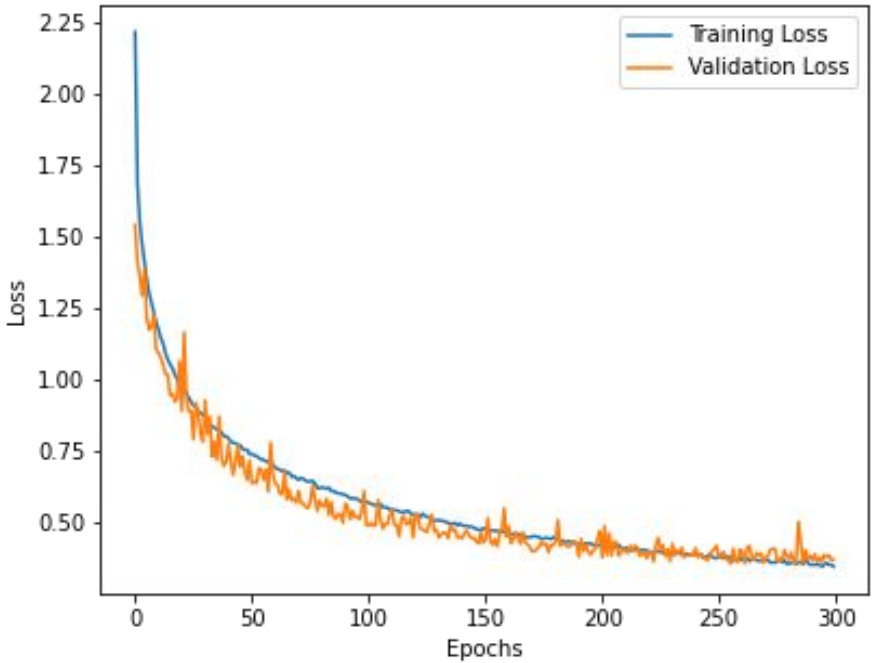
Confusion matrix





Learning curve





Conclusions

Conclusions

- CNN performs better than ANN for image data
- Max pooling layers reduce the growth in parameter numbers when adding more layers to CNN
- Dropout layer reduced the over-fitting on training data
- Batch normalisation standardises input and stabilises learning process
- Visualisation of the feature maps of the CNN demonstrates deeper layers learn higher order features and more feature maps show no activation for a given image instance
- We were able to get relatively good training results with a very small number of epochs, but to prevent over-fit and to get to a truly high accuracy, we had to not only apply additional techniques and layers, but also increase significantly number of epochs
- Creating a well performing model requires a lot of skill as well as time and resources for the model to be trained

References

References

- Madhugiri, D. (2021) *Using CNN for image classification on CIFAR-10 Dataset*, *Medium*. Available at: https://devashree-madhugiri.medium.com/using-cnn-for-image-classification-on-cifar-10-dataset-7803d9f3b983 (Accessed: 22 June 2023).
- Garg, A. (2022) *How to classify the images of the CIFAR-10 dataset using CNN?*, *Analytics Vidhya*. Available at: https://www.analyticsvidhya.com/blog/2022/09/how-to-classify-the-images-of-the-cifar-10-dataset-using-cnn/ (Accessed: 22 June 2023).
- Le, J. (2018) *The 4 convolutional neural network models that can classify your fashion images*, *Medium*. Available at: https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d (Accessed: 22 June 2023).
- Kubat, M. (2021) *Introduction to machine learning*. Cham: Springer.
- Chollet, F. (2021) 'Getting started with neural networks', in *Deep learning with Python*. Shelter Island: Manning Publications, pp. 56–92.
- *Implementing a CNN in Tensorflow & Keras* (2023) *LearnOpenCV*. Available at: https://learnopencv.com/implementing-cnn-tensorflow-keras/ (Accessed: 11 July 2023).
- Chollet, F., 2021. *Deep learning with Python*. Simon and Schuster.
- Kingma, D.P. and Ba, J., 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.