NEURAL NETWORK MODELS

FOR OBJECT RECOGNITION

Maria Ingold
12693772
Unit 11
Machine Learning
University of Essex Online
20 January 2025

CONTENTS

INTRODUCTION

Welcome to Neural Network Models for Object Recognition which evaluates convolutional neural networks with the CIFAR-10 dataset. I'm Maria Ingold, a Master of Science student in Artificial Intelligence and industry executive in media technology.

Application

In media, companies like Eluvio (N.D.) provide object recognition services for film and television content. The query, 'fighting in Bullet Train' in Figure 1, shows it could include action, celebrity, or landmark recognition. Object recognition enables the creation of descriptive text, or metadata, which can be used to create time-coded tags. Tagging is a time-consuming manual process, so automation using machine learning creates new opportunities like monetising archive on selected topics.

On the live production side, Pixellot's AI-automated sports camera automatically follows the action using ball and player tracking. The object recognition was trained using Convolutional Neural Networks on over 5 million games across a range of sports (Martin, 2018; Bickerton, 2024). This tracking further enables real-time player analytics, helping kids, niche sports, and professional leagues automatically stream matches while improving their game (Pixellot, 2024).

IMAGE DATA

CIFAR-10

Object recognition requires objects to recognise and a model to recognise them. As a classification problem it maps its solution to a finite set of values, called labels or classes (Russell & Norvig, 2021).

The CIFAR-10 dataset has 60,000 colour images of 10 classes—aeroplane, automobile, bird, cat, and so on (Krizhevsky, N.D.). There are 6,000 images per class, so 6,000 cats, 6,000 dogs, 6,000 frogs. Figure 3 shows a random sample of the 10 classes.

This presentation illustrates object recognition using this CIFAR-10 dataset to train a Convolutional Neural Network, or CNN, to determine the probability that an image is in a certain class (Figure 4). A CNN is a form of deep learning and is represented mathematically in code as a model (Kubat, 2021). The accuracy and loss of the model indicates the likelihood that it will see a cat as a cat, instead of a dog. As models are designed and iteratively improved, there are a range of pitfalls and learnings to avoid and understand along the way.

Data Loading

CIFAR-10 was imported and loaded from the TensorFlow Keras dataset in Figure 5. TensorFlow is a Python programming language platform for machine learning (TensorFlow, N.D.c). Keras is its Application Programming Interface, or API (TensorFlow, N.D.a). Each image (referred to as x) has a label (referred to as y).

Each label in CIFAR-10 is a number 0 through 9, which corresponds to the LABEL_NAMES depicted here. 0 for ‘airplane’ and 9 for ‘truck’. LABEL_NAMES makes it easier for humans to read.

Partitioning

Figure 6 shows the CIFAR-10 dataset includes 50,000 images for training and 10,000 for test (Krizhevsky, 2009). Kubat (2021) recommends setting aside 10-20% of the training set to create a validation set. The training set trains candidate models. The validation set evaluates them, preventing overfitting and influencing. And the test set is only used once to perform an unbiased test of the best model on unseen data.

I made two mistakes, however. The first was not initially specifying a random state to ensure a reproducible split, essential for comparing models across sessions.

Stratification

The second, which remained unfixed due to long model training times, is adding stratification (scikit-learn, N.D.). By not specifying stratification, I slightly imbalanced the class distributions in the training and validation sets (Figure 7). These should be 4000 samples for training and 1000 for validation, respectively. While small variations, this under and over-representation could account for slight bias, making certain classes harder to predict, as seen later.

Preprocessing: Normalisation

To be used most effectively by the neural network, the data is pre-processed. In this case, the *image* data is normalised, and the *label* data is one-hot encoded. Colour pixels in images are composed of red, green, and blue, or RGB. As shown in Figure 8's (W3 Schools, N.D.) calculator, each ranges from 0 to 255. 0 is black, or off, and 255 is at full intensity.

Figure 9 shows that the 100th image in the training set is a truck. This is its red channel. Prior to normalisation it has a range of 0 to 255.

After normalisation, Figure 10 shows everything remains the same except peak intensity is now 1. To stay similar, all three—train, validate, and test—need to be normalised (Kubat, 2021). Normalising features to one scale helps algorithms converge faster, prevents feature dominance, and simplifies comparisons (Machine Learning Models, N.D.).

Preprocessing: One-Hot Encoding

As seen in Table 1, all one-hot encoding vector values are 0, except the one corresponding to the *label*. The 100th image, a truck, is label 9, just represented differently after one-hot encoding.

CNNs require numerical input and categorical data like the nominals 0, 1, 2, 3, could be misconstrued to have rank which would introduce bias (Gu & Sung, 2021; GeeksForGeeks, 2024). One-hot encoding is required to classify correctly.

CONVOLUTIONAL NEURAL NETWORKS

Machine Learning Architecture

Now that we have the images, we need a model to recognise them. Figure 11 shows artificial neural networks (ANNs) are a subset of machine learning. ANNs contain deep learning which includes CNNs (Taye, 2023). As Figure 12 shows, Kubat (2021) describes deep learning as the more modern approach.

Neural Network Architecture

At its simplest, Figure 13 shows the artificial neuron, or perceptron, takes an input feature (x), applies a weight, or importance, (w), and passes a weighted sum of those inputs to an activation function (f(x)) (Ansari, 2023). Then the activation function predicts the output label (y) for the input image, like ‘cat’.

Deep Learning Architecture

Grouping neurons creates a neural net, known as a multi-layer perceptron, or MLP. This is also referred to as deep learning (Ansari, 2023). Figure 14’s MLP contains three layer types: input, hidden, and output. Hidden layers provide learning.

CNN Architecture

ANNs require manual feature engineering, while CNNs automatically extract features from input images. Figure 15 shows feature maps from a convolution layer are input into a fully connected multilayer perceptron, or FC (Ansari, 2023).

CNN Architecture: Detail

CNNs are more efficient with fewer parameters and more scalable than ANNs. Figure 16 shows a simplified CNN architecture (Gupta, 2017; Ansari, 2023; Kromydas, 2023). A 32x32 RGB colour image is input. The convolution layer applies a 3x3 filter, or kernel, and then a Rectified Linear Unit, or ReLU, activation function. Max pooling then summarises the features and reduces the feature map by half, helping mitigate overfitting and computational complexity. This output can be input into another convolutional block. Once all the convolutions are complete, the feature map is flattened to a 1-dimensional vector. This is then fully connected to the MLP, or dense layer, and combined with Softmax to classify. Softmax normalises the output to [0, 1] so a probability can be assigned to each class label.

Real-world scenarios, like recognising which Chris is on screen in an Avenger's film, require more layers, but that adds computational complexity resulting in longer computation times. While max pooling helps, we can also drop neurons. All of which will be explored in the model designs.

MODELS

Model 1

Model 1 Architecture

Figure 17, adapted from Kromydas (2023), shows three convolutional blocks, two with max pooling. It starts with 32 3x3 filters and increases to 64 then 128 to enable extraction of more complex features. At the end of the last convolutional block, the number of channels in the activation map is 128.

After flattening into a 1-dimensional vector, the features are transformed into class probabilities in the fully connected (dense) layers. Dropout helps with overfitting by randomly dropping 50% of the neurons. In retrospect, I would probably leave this out of the baseline model and add it in later. Softmax's normalisation ensures the probability corresponds to class labels. Softmax and ReLU are both activation functions.

Activation Functions: ReLU

Figures 18 and 19 show ReLU outputs x if x is greater than 0, but 0 otherwise (Gupta, 2017). ReLU helps network convergence and works well with pixels which do not have negative values (Ansari, 2023). In contrast to Sigmoid, ReLU does not suffer from the vanishing gradient, which means when adding more than three hidden layers the model remains computationally efficient and can add further value (Kubat, 2021).

Activation Functions: Softmax

Figure 20 and Table 2 show a Softmax probability output scenario that might result in 83% probability that the image was a cat. The probabilities sum to 1 (Ansari, 2023).

Loss Function

The categorical cross-entropy loss function predicts more than two classes (Ansari, 2023; GeeksForGeeks, 2024a). It calculates the difference between predicted labels and true labels so that the model can generate more accurate predictions. In Figure 21 the Softmax probability of 83% likelihood of being a cat, is close to the true value of 1. Here, categorical cross-entropy is small at 0.18. However, if a dog had been incorrectly predicted, the loss would be higher, requiring the model to adjust its weights.

Note that in compile is also where the learning rate optimiser Adam is set, with a learning rate of 0.001.

Epochs

In model training, epochs, or training iterations, are set as in Figure 22. As the number of weights increase, the number of epochs required for convergence also increases, which means computation time and costs increase (Kubat, 2021).

My third mistake was not realising that each code run can deliver slightly different results. To introduce further reproducibility, a model checkpoint has been added to store the best model for future comparison and potential transfer learning.

The best model is determined by a continual improvement to loss, because the primary goal of the validation set is to minimise the loss function (Russell & Norvig, 2021). The secondary goal is to maximise accuracy.

Batch size, also set here, and the learning rate set in compile are fixed respectively at 64 and 0.001 across all models and offer future opportunities for improvement.

Model 1 Accuracy and Loss

Validation accuracy and loss can be compared to training accuracy and loss (Russell & Norvig, 2021). If the training performed well, but the validation did not, that indicates overfitting. The divergence in Figure 23 shows the model stops generalising at much lower than 20 epochs. The further epochs are not useful, and the model is overfitting. Overfitting means it is paying too much attention the training data.

Model 2

Data Augmentation

Model 2 aims to address overfitting by adding two things. It introduces early stopping and data augmentation. Early stopping stops after 3 epochs of no improvement and reverts to the best weights.

Adding more training data using data augmentation generates new images by transforming existing ones. The cat in Figure 24 is randomly shifted and flipped in Figure 25. Other transformation types could be explored in the future.

As a note, adding a subsequent model to the same file is where I discovered my fourth mistake, which was that I needed to clear the runtime, to ensure training the previous model did not affect the current one.

Model 2 Accuracy and Loss

Figure 26 shows that with early stopping, it stopped at epoch 18, with no divergence between training and validation. Training has slightly worse accuracy and loss due to the additional complexity from data augmentation, but the validation has improved and consistently exceeds training both for accuracy and loss. Overfitting appears to be mitigated, and generalisation improved.

Model 3

Model 3: L2 Regularisation

It was all going well until Models 3 and 4. My fifth and sixth mistakes were because I didn't understand a technology. TensorFlow (N.D.b) mentioned that combining dropout (in since Model 1) with L2 created the best model to combat overfitting. So, I added L2 regularisation. Everything else remained the same, as seen in Figure 27.

Model 3 Accuracy and Loss

Figure 28 shows training and validation accuracy plunged, and loss worsened. Validation became more erratic. It was now underfitting.

Model 4

Model 4 Accuracy and Loss

So, in Model 4, L2 was lowered from 0.01 to 0.001. However, Figure 29 shows that it still performed badly. The only clarity I have discovered is from Google For Developers (N.D.) training module. It suggests that one, early stopping may not be an optimal form of regularisation and, two, that learning and regularisation rates pull weights in opposing directions. Regularisation pulls towards zero, while learning pulls away. Further balance could be explored in the future.

Model 5

Model 5 Architecture

Model 5 removed L2, going back to Model 2 as a base. It then took inspiration from the Zhang (2021) discussion on AlexNet and added an extra convolution and pooling layer as seen in Figure 30. With 256 filters, extraction of more complex features is possible. Epochs were increased to 30, just in case.

Model 5 Accuracy and Loss

Figure 31 shows Model 5 has the best accuracy and loss out of the 5 models.

Model Comparison

Table 3 compares the models. The baseline model included dropout, which may have been better to add at a second stage for comparison. Data augmentation and early stopping help mitigate overfitting, however, L2 regulation underfits at 0.01 and 0.001. This demonstrates that even a small L2 penalty harms the model's learning potential. Model 5 removes L2 and adds a convolution layer with 256 filters and a pooling layer. Model 5 is slightly better than Model 2. Overall, the greatest impact was from data augmentation and the worst from L2 equals 0.01.

TEST

Testing

Every model saved its best result. All are in in GitHub under models. Testing loads Model 5. Its test accuracy is 0.73, on par with Model 5's validation, while its test loss is 0.79, slightly higher. Looking at images and their label predictions clarifies why the model makes mistakes. Figure 32 shows the lack of clarity of the cat could confuse it with a dog, while the ship has sky behind it, leading to confusion with an aeroplane. With almost perfect prediction, however, the aeroplane includes sky and a runway.

Insights

Figure 34 shows there are 1000 images per label, so perfect prediction in Figure 33's confusion matrix would be 1000. However, the best prediction on the true positive diagonal was automobile at 900 with 71 false negatives predicting a truck instead. Ship was predicted to be an automobile with 73 false positives. Birds were the worst, with less than half true positive. They were confused most with frogs, but also with all the other animals and aeroplanes. Cats and dogs were also confused. As were deer and horses. And ships and aeroplanes, as seen in the previous slide.

Figure 34's classification report shows the model does well at classifying ships with the highest precision and F1-score, whereas bird has the lowest recall. Overall, this model's accuracy correctly classifies 73% of test images.

Further data augmentation on the weak classes could provide additional data to help better differentiate between say, a boat with a sky background and an aeroplane in the sky, small furry cats and dogs, and horses and deer with four long legs.

REFLECTION AND CONCLUSION

I made a range of mistakes which have translated into the learnings in Table 4.

- Because models change per run, fix random values to improve reproducibility.
- Add stratification if splitting from a balanced training set.
- Keep the baseline simple so effects from additions like dropout are clearer.
- Make one change per model to see the effect. But they take a while to run.
- So, store the best model and investigate transfer learning.
- Plan for long training times, use a GPU, but realise Colab has usage limits.
- Clear the runtime between model training to avoid retraining.
- Adding something, like L2, may not improve the model, especially if it conflicts with something else.
- 73% model accuracy doesn't mean every class performs at 73%.
- And finally, there is always room to grow! I'd like to explore further data augmentation for weak classes, batch normalisation, and dynamic learning rate adjustment.

If you want more, my code and models are on GitHub.

I've realised how crucial object recognition is to media, sports and entertainment. Eluvio needs highly accurate predictions with minimal errors, although video on demand content, can use a human in the loop. However, Pixellot needs live accuracy to track balls and players. I also now understand how Amazon Prime Video’s X-Ray works in Figure 35 (Jarrett, 2022). Thanks to investigating the use of Convolutional Neural Networks with CIFAR-10 I now have a greater appreciation for the effort required to deliver these services.

REFERENCES

Ansari, S. (2023) *Building Computer Vision Applications Using Artificial Neural Networks: With Step-by-Step Examples in OpenCV and TensorFlow with Python*. 1st ed. Apress (Springer Nature). Available from: https://learning.oreilly.com/library/view/building-computer-vision/9781484298664/html/493065_2_En_5_Chapter.xhtml [Accessed 14 January 2025].

Bickerton, J. (2024) *Pixellot updates AI-driven sports camera*. Available from: https://www.broadcastnow.co.uk/production-and-post/pixellot-updates-ai-driven-sports-camera/5197703.article [Accessed 12 January 2025].

Eluvio (N.D.) *AI/ML Tagging | Eluvio Documentation*. Available from: https://docs.eluv.io/docs/content/ai-ml-tagger/ [Accessed 12 January 2025].

GeeksForGeeks (2024a) *Categorical Cross-Entropy in Multi-Class Classification*. Available from: https://www.geeksforgeeks.org/categorical-cross-entropy-in-multi-class-classification/ [Accessed 16 January 2025].

GeeksForGeeks (2024b) *One Hot Encoding in Machine Learning*. Available from: https://www.geeksforgeeks.org/ml-one-hot-encoding/ [Accessed 13 January 2025].

Google For Developers (N.D.) *Overfitting: L2 regularization | Machine Learning*. Available at: https://developers.google.com/machine-learning/crash-course/overfitting/regularization [Accessed 16 January 2025].

Gu, B. & Sung, Y. (2021) Enhanced Reinforcement Learning Method Combining One-Hot Encoding-Based Vectors for CNN-Based Alternative High-Level Decisions, *Applied Sciences 2021, Vol 11, Page 1291* 11(3): 1291. DOI: https://doi.org/10.3390/APP11031291.

Gupta, V. (2017) *Understanding Feedforward Neural Networks | LearnOpenCV*. Available from: https://learnopencv.com/understanding-feedforward-neural-networks/ [Accessed 15 January 2025].

Jarrett, C. (2022) *What is X-Ray on Prime Video? How to view bonus content*. Available from: https://www.aboutamazon.com/news/entertainment/what-is-x-ray-on-prime-video [Accessed 16 January 2025].

Krizhevsky, A. (2009) Learning Multiple Layers of Features from Tiny Images. Available from: https://api.semanticscholar.org/CorpusID:18268744 [Accessed 2 January 2025].

Krizhevsky, A. (N.D.) *CIFAR-10 and CIFAR-100 datasets*. Available from: https://www.cs.toronto.edu/~kriz/cifar.html [Accessed 12 January 2025].

Kromydas, B. (2023) *Convolutional Neural Network: A Complete Guide*. Available from: https://learnopencv.com/understanding-convolutional-neural-networks-cnn/ [Accessed 15 January 2025].

Kubat, M. (2021) *An Introduction to Machine Learning*, *An Introduction to Machine Learning*. Springer International Publishing. DOI: https://doi.org/10.1007/978-3-030-81935-4/COVER.

Machine Learning Models (N.D.) *The Importance of Data Normalization in Machine Learning*. Available from: https://machinelearningmodels.org/the-importance-of-data-normalization-in-machine-learning/ [Accessed 13 January 2025].

Martin, S. (2018) *Pixellot Scores a Goal for Amateur Sports by Automating Video Analysis with AI | NVIDIA Blog*, *NVIDIA*. Available from: https://blogs.nvidia.com/blog/pixellot-gpu-amateur-sports-video-analysis-ai-soccer/ [Accessed 12 January 2025].

Pixellot (2024) *Unveil the Future of Sports Production: What to Expect From Pixellot at IBC 2024?*, *LinkedIn*. Available from: https://www.linkedin.com/pulse/unveil-future-sports-production-what-expect-from-pixellot-msuof/ [Accessed 12 January 2025].

Russell, S. & Norvig, P. (2021) *Artificial Intelligence: A Modern Approach, Global Edition*. 4th ed. Pearson Education, Limited.

scikit-learn (N.D.) *train_test_split — scikit-learn 1.6.1 documentation*. Available from: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html [Accessed 13 January 2025].

Taye, M.M. (2023) Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions, *Computation 2023, Vol 11, Page 52* 11(3): 52. DOI: https://doi.org/10.3390/COMPUTATION11030052.

TensorFlow (N.D.a) *Keras: The high-level API for TensorFlow | TensorFlow Core*. Available at: https://www.tensorflow.org/guide/keras [Accessed 12 January 2025].

TensorFlow (N.D.b) *Overfit and underfit | TensorFlow Core*. Available from: https://www.tensorflow.org/tutorials/keras/overfit_and_underfit [Accessed 16 January 2025].

TensorFlow (N.D.c) *TensorFlow basics | TensorFlow Core*. Available from: https://www.tensorflow.org/guide/basics [Accessed 12 January 2025].

W3 Schools (N.D.) *Colors RGB and RGBA*. Available from: https://www.w3schools.com/colors/colors_rgb.asp [Accessed 13 January 2025].

Zhang, X. (2021) The AlexNet, LeNet-5 and VGG NET applied to CIFAR-10, *Proceedings - 2021 2nd International Conference on Big Data and Artificial Intelligence and Software Engineering, ICBASE 2021* 414–419. DOI: https://doi.org/10.1109/ICBASE53849.2021.00083.