Enhancing Karooooo's AI Dashcams with Convolution Neural Networks

Matthew Bowyer – UOEO

Good day, executives and colleagues. As some of you may know, my name is Matthew Bowyer—your R&D analyst as well as a master's student at the University of Essex Online. Today I am presenting a research project that provides a practical example of the developments we can make to our AI-powered cameras. From our current generic "object ahead" system, I want to improve our camera logic, which will, in turn, improve safety and usability.

Karooooo (Yes, we not only always spell it with 5 o's but we have the rights to 6,7,8,9,10 and 11 o's) Karooooo cameras excel at monitoring driver behaviour and proximity alerts (Cartrack, n.d.) but we can do more. Our cameras lack granular object awareness. If our current camera detects any object, it will help the driver avoid bumper bashes by letting the driver know verbally that an object is ahead. But what if the object is an animal, a pothole, or a truck? Then "object ahead" helps but the lack of description could cost the driver critical seconds trying to identify the threat rather than avoiding it. I am going to show that models can be trained to tell exactly what is ahead. By improving the detail given to the driver, if a dog is in the distance ahead, the driver should be notified by the camera, "Dog ahead" rather than "Object ahead".

To train this model, I used CIFAR-10 (Krizhevsky, 2009). Which includes 60 000 colour images across 10 classes. As you can see, 3 random examples of images and their labels from the dataset. A frog, a truck and another truck. This dataset was chosen as it's small enough for prototyping but complex as it contains 10 different classes the model needs to learn to differentiate between. This includes the examples I have shown as well as birds, dogs, ships and more.

Getting into details, I cannot just give a model pictures and labels and hope it works. Some work needs to be done on the data. First, as you can see in the pie chart, I split the 60 000 images into training, testing and validation data. In a paper by Jeremy Jordan (Jordan, 2017) on evaluating machine learning models, he explained a typical random split is to use 60% of the data for training, 20% for validation and 20% for testing. This means that the model sees the training data and trains on it, every lesson (Called epochs) the model goes through the test data, and the model then gets to see the validation data but has to tell what the labels are. Finally, after all epochs, the model gets tested on the test data. Giving the model too much training data can lead to overfitting which means the model is very good at its training data but not good at the test or validation data. Giving the model too little training data can lead to underfitting which means the model is not trained well enough to be good at predicting what the image label is. For my models, I chose a 75%, 8,3% and 16,7% split. This was because of the large number of classes, I felt the model would need a bit more training data to make sure all classes were represented well. The downside is that the larger training data could cause overfitting and poor results after testing.

To improve the model's performance. I had to apply various techniques like normalisation, one-hot encoding and data augmentation.

Normalisation – The pixel values of the images come in a range of 0 to 255. This gives the colour of the image. I normalise them between 0 and 1 to improve efficiency and readability for the model. (Jlassi & Dixon, 2024) The disadvantage to normalisation is that it takes extra computation to run through the dataset. The advantages are that normalisation does not change the fit of the pixels meaning the change won't cause bias on the outcome of the model, as shown in the two graphs.

One-hot encoding – This converts the table into binary (1 and 0) vectors. Which is easier for computers to read through. For example, if you had a gender column which identified each user as one of two genders, either Male or Female. One-hot encoding will remove the gender column and add a column for each unique category in gender, in this case two, then the user who identifies with Male will have 1 under the male column and 0 under the female column. The disadvantages are that this takes processing power before the model has even started training and only works on certain types of models.

Data augmentation includes resizing images to potentially strengthen the learning rate by improving the variability in the images. This can also lessen the chances of overfitting as the dataset now has more variability. The disadvantage to using a single data augmentation method is that the real world has multiple factors including brightness changes and flipped objects, which are some of the additional data augmentation methods that could be included in model creation. Which can create models which have higher real-world accuracy.

CNNs (Convolution Neural Networks) are one of the most widely used models today. They work by processing grid-like structures through one or more layers of neural networks, which break down the structure of the images for processing. (Geeksforgeeks, 2024). Digital images are grid-like structures and CNNs specialise in processing grid-like structures. (Mishra, 2020). The disadvantages include that the CNNs will not be able to tell where the object is exactly or what orientation the object is. (Geeksforgeeks, 2024). This is why, for this project, I did not choose classic machine learning or advanced machine learning. Deep learning using CNNs is a proven method for image recognition.

I built 3 CNNs to compare performance, costs and overall usability at Karooooo. Model 1 is a simple model I made. Model 2 is a model with a base built by a Google-developed model called MobileNetV2. The third model is a popular powerful model called ResNet-50.

Models 2 and 3 use a method called transfer learning. A method of using pre-built models for similar purposes as the base for the models for this project. This is common practice and is used to boost the results of the model. (Gupta et al, 2022). Think of it as model 1 learns only from the database we gave it, while model 2 and 3 start with a base that has already seen thousands to millions of everyday images. This way model 2 and 3 may already be able to differentiate between the images we now give in the dataset well from the start.

All models use RGB images, a 32x32 input layer. This is how I input the data. The models use Adam optimiser and categorical cross entropy, ReLU in all hidden layers and a final dense layer activated by SoftMax, which are widely used hyperparameters to compile models (Sharma, 2017). Hyperparameters are the models' settings. Models work by inputting data, processing it through the CNN layers and then outputting the image label that the model processed to most likely be true. Models are not just one big artificial brain. They are several specialists all doing their own job to achieve a specific goal. Some specialists, using data augmentation discussed earlier, they learn how

to handle rotated, resized and brightened versions of the images to be able to recognize the images, even when conditions change. Other layers look for patterns and textures like fur, road markings or metal. Finally, the output experts come together and based on what they understand from the previous layers, they give an educated prediction on what they think it is.

Choosing the same widely used compilers for all models is common practice as they are proven methods and make the models directly comparable. The perfect model cannot be built, but this method gives us a great base to work off of when we dive into testing the CNNs on Karooooo data. We can then attempt other compiler layouts, hyperparameters and bases. The downside is the models have similar features, meaning I am not exploring all possible model designs. However, it would not be possible to test every type of model variation. It is possible to test many of them and refine models, I will dive into this later when we discuss the roadmap for this project.

The models differ in the number of epochs. Model 1 has 30, model 2 has 25 and model 3 has 20. Respectively, the models have 800 000, 2,3 million and 23 million trainable parameters. These are the weights and biases the model changes throughout learning. These parameter figures come from the different layers I chose in each model. All models had convolution layers, which detect patterns and textures. All models also had Dense layers at the end which is the layer that makes the final decision. Model 2 and 3 had pre trained layers, I added a dropout layer which prevent overfitting by "Shutting down" a certain percentage of the neurons during training. As models grow in depth and complexity, so do the parameters. Model 1 uses a batch size of 128, model 2 uses a batch size of 96 and model 3 uses a batch size of 64. Batch size is the number of images processed at a time. I lessen batches as the models become more complex, this way each epoch uses less memory in the bigger models. Models 1 and 2 use a learning rate of 0,001 and model 3 uses a learning rate of 0,0005. Learning rate controls updates to the model throughout each batch. The higher the learning rate the bigger jumps the model will make. Since model 3 has a smaller batch size, I made the learning rate smaller to avoid instability and large updates throughout training.

In summary, model 1 is the smallest, simplest and quickest model to train. Model 2 is significantly bigger, with transfer learning and takes a longer time to train. Model 3 is the biggest, also with transfer learning and takes the longest time to train. On paper, each level of the model should provide better results than the lower level of the model.

Time for the big reveal. Which model did best? What sort of accuracy can we get the test data to? Let's start with the big one: accuracy, the percentage of total correct predictions made by each model. Rounded up, Model 1 had 69% accuracy, Model 2 had 81% accuracy and Model 3 had 32% accuracy. Model 2 and 3 used transfer learning, which as discussed means before they were trained on our dataset, the models saw millions of images and trained on them. Model 3's low accuracy compared to model 2 could be due to model 2 having been trained on images similar to our dataset. I do not see any obvious signs of overfitting as model 3 does badly on the training data as well as the test data. My assumption is the images model 3 has been trained on must not compliment our dataset.

We see similar percentages for Precision, recall and F1-score. Which all just help make sure the model is not just a random label generator. Precision shows how often the model is right when it says "yes", Recall shows how well the model catches all true cases and F1-score the harmonic mean of precision and recall.

MCC (Matthews Correlation Coefficient), is a single metric that considers all 4 categories of a confusion matrix. A confusion matrix is used to show each class, how many times it was correctly predicted and how many times other classes were wrongly predicted. We can see the confusion matrix for model 2, the best-performing model. The most accurate class is the truck with 913 true-positive predictions, positively ironic. The worst performing class is the cat, due to the cat and dog getting mixed up. With a dog being selected 141 times when the image was a cat. This is where we can decide to merge animals into one category to avoid false positives.

Using MCC we can see between -1 and +1 if the model is guessing (a score of zero) or if the model is perfect at predictions (1) or perfect at predicting the wrong answer (-1). The MCC for the models were 0.66, 0.78 and 0.25. Model 3 is almost as close to being categorized as a random guessing model as model 2 is to being categorized as a perfect predictor.

Diving deeper into model 2. We can look at the Validation and training accuracy and loss over the epochs. We can see the accuracy at the 1st epoch already started at roughly 75% accuracy for the training data and 79% for the validation data. This signifies that the training data is well rounded as the model does well on the validation data too. This also emphasizes that the method of transfer learning shows model 2's training before I used it compliments our dataset very well. Looking at the right graph. Loss shows how different the guess is from the actual image label. We can see that loss is decreasing as the difference to the actual image gets smaller. This means the training went very well. The graphs seem to be close to their peak of around 82%, signifying that more epochs would not improve accuracy. Some changes will need to be made to the compilers if we want a better model.

Model 2's 81% accuracy and near-perfect MCC score outshine models 1 and 3. This was a practical example. Imagine what we could do with the green light on this project. The accuracy we could work up to, the inference time we could lower and the lives we could save.

The practical example is a success. The next step is to integrate the model with Karooooo's data. Fix issues like merging all types of animals into one class to avoid the cat-dog false positive issue. Merging class automobiles and class trucks for the same reason. We also need to make sure that we have images of rare hazards such as cars with no lights on or animals on the road at night. These outlier hazards are important to not overfit the models with common data. Increased data augmentation will also help in this situation. Further operational considerations include avoiding too many alerts or false alerts, so as not to overwhelm the driver. We can achieve this by only providing a warning if the model is 95% or more certain of what it has detected. Development into these key factors can skyrocket the accuracy and rollout Karooooo's new AI camera safety features.

With datasets being built internally, models being trained on our own and backed on our servers. Costs can be kept to a minimum as we will be producing and training the models in-house. Model 2 with 81% test accuracy is lightweight in terms of power usage and a great start to this project.

By enabling object-specific alerts, the advanced system can reduce rear-end, animal and pothole collisions, and doing so with minimal additional power drawn by the cameras. This is a step into the future of safety.

Thank you

References

Cartrack. (Unknown) AI-powered dashcams and live stream camera technology. Available from: https://www.cartrack.co.za/platform/features/safety/ai-powered-cameras. [Accessed 9 July 2025]

Krizhevsky, A. (2009) The CIFAR-10 dataset. Available from: https://www.cs.toronto.edu/kriz/cifar.html#::text=CIFAR-10%20python%20version. [Accessed 13 July 2025]

Jordan, J (2017) Evaluating a machine learning model. Available from: https://www.jeremyjordan.me/evaluating-a-machine-learning-model/. [Accessed 11 July 2025]

Jlassi, O., Dixon, PC. (2024) The effect of time normalisation and biomechanical signal processing techniques of ground reaction force curves on deep-learning model performance. Available from: https://www.sciencedirect.com/science/article/pii/S0021929024001945. [Accessed 10 July 2025]

Geeksforgeeks. (2024) Difference between ANN, CNN and RNN. Available from: https://www.geeksforgeeks.org/deep-learning/difference-between-ann-cnn-and-rnn/. [Accessed 15 July 2025]

Mishra, Mayank. (2020) Convolutional Neural Networks, Explained. Available from: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939/. [Accessed 15 July 2025]

Gupta, J. et al. (2022) Deep Learning (CNN) and Transfer Learning: A Review. Available from: https://iopscience.iop.org/article/10.1088/1742-6596/2273/1/012029/pdf. [Accessed 19 July 2025]

Sharma, A. (2017) Convolutional Neural Networks in Python with Keras. Available from: https://www.datacamp.com/tutorial/convolutional-neural-networks-python. [Accessed 11 July 2025]