**Slide 1: Title Slide**

Hello and welcome to this presentation on neural network models for object recognition. I am excited to show you the process of designing and evaluating a neural network for object recognition using the CIFAR-10 dataset (Krizhevsky, 2009). We will discuss how important object recognition is and how it can be used, and then explain the techniques used to prepare the data and any insights we can gather on the dataset. I will then walk you through the model’s architecture, training and evaluation. Finally, we will reflect on any challenges faced during the course of the project and delve into future improvements that could be made.

Every aspect of this neural network model was designed in Python, including any visualisations, using libraries such as: Matplotlib, Pandas and TensorFlow.

**Slide 2: Introduction to Object Recognition**

Object recognition is a crucial aspect of artificial intelligence and computer vision, which allows machines to identify and classify objects within images or videos. This ability is widely utilised in many industries such as autonomous vehicles, surveillance, healthcare and education. In the motor industry, this has allowed the creation of driverless cars, due to the ability to use object recognition to detect people, signs, cars and other obstacles, to ensure safe driving decisions. In security, suspicious activity and potential threats can be neutralised, due to the likes of facial recognition, which enhances public safety. Furthermore, in hospitals, object recognition can be used to analyse medical images, which can assist doctors in early diagnosis and treatment planning.

Significant improvements in object recognition have been made from the development of sophisticated deep learning algorithms, which has only been made possible due to the availability of larger datasets and improvements in computational power over the years. Despite these improvements, there are still challenges to be faced in object recognition, such as handling variations in lighting, angles and occlusions within images. We require neural networks to be well-designed and capable of learning powerful features from complex data, in order to overcome these challenges.

**Slide 3: About the CIFAR-10 Dataset**

The CIFAR-10 dataset (Krizhevsky, 2009) consists of 60,000 colour images, 32 by 32 pixels wide, categorised into 10 distinct classes, and is one of the most used benchmark datasets for classifying images. The classes consist of some vehicles: airplanes, automobiles, ships and trucks; then also some animals: birds, cats, deer, dogs, frogs and horses. The dataset is already split into 50,000 training and 10,000 test images, which provides us with an effective basis to train and evaluate our model.

One issue we do face with this dataset is the low resolution of the images, which makes it very hard to distinguish between similar objects, such as cats and dogs, or objects with similar characteristics, such as the wheels on cars, trucks and planes, despite their differences. The images also contain variations in lighting, backgrounds and orientation, making it vital to include robust preprocessing and augmentation techniques to enhance the model’s ability to generalise to unseen data.

**Slide 4: Data Preparation and Preprocessing**

Before we train the model, we must undergo several preparation steps to ensure optimal performance. Firstly, we need to make sure we have data for training, validation and testing; now we already have a five to one split of training and test data, but we do not have the validation data. We split 10,000 objects from our training data to set aside as validation data, leaving two-thirds of the data for training and the remaining sixths for testing and validation, as you can see in the pie chart. The training set was used to fit the model, while the validation set helped fine-tune hyperparameters and detect overfitting, and the test set provided an unbiased evaluation of the final model.

**Slide 5: Data Augmentation**

We normalised the images to scale the pixel values between 0 and 1, which helped improve numerical stability and convergence speed during training. Furthermore, we used data augmentation (Awan, 2024) to expand the dataset, using the different variations of images we could create, using various techniques. These techniques consisted of random rotations, horizontal flipping, and zooming, as seen in the example images. These augmentations helped the model become more robust by learning to recognise objects from different perspectives and conditions.

**Slide 6: Model Architecture Overview**

The neural network architecture used in this project follows a convolutional neural network (CNN) design, which is well-suited for image classification. The model consists of three main stages: feature extraction, feature transformation, and classification.

In the feature extraction stage, the model uses convolutional layers followed by batch normalisation and max-pooling layers. The convolutional layers help to identify patterns such as edges and textures within the images, while batch normalisation standardises activations and max pooling reduces spatial dimensions, making for more efficient computations.

Next, we look at the feature transformation stage, where the extracted features are flattened and passed through dense, fully connected layers. Dropout layers were also included to reduce overfitting by randomly deactivating neurons during training.

Finally, in the classification stage, a dense output layer with a SoftMax activation function (Gómez, 2018) generates class probabilities for each of the 10 categories.

**Slide 7: Activation Functions**

Activation functions allow our neural network to learn complex relationships by using a non-linear approach. We selected the Rectified Linear Unit (ReLU) for hidden layers in our model, due to its efficiency in computing and ability to set all negative values to zero, preventing vanishing gradients.

We then used the SoftMax activation function (Gómez, 2018) to convert the outputs from the model into probability distributions across the 10 classes, ensuring that the sum of probabilities equal one. This is ideal for multi-class classification tasks.

We can compare ReLU with alternative activation functions such as Sigmoid and Tanh, as you can see in the graphs. The outputs in the graph show us ReLU provided much faster training times and better performance for deeper networks, compared to the other activation functions.

**Slide 8:** **Loss Function and Optimiser**

The model we used was trained using categorical cross-entropy (Gómez, 2018) as the loss function, which measures the difference between predicted and actual class labels. This loss function penalises incorrect predictions proportionally to their confidence level, which makes it particularly effective for multi-class classification tasks like this. As you can see in the formula shown on the slide, the function sums up the negative log probabilities across all training samples to do this.

We also applied learning rate decay (Hugging Face, n.d.) to the optimisation process, which can be explained easily by looking at the line graph. Initially we start with a high learning rate, which allows for larger weight updates during the early stages of training. However, as the model converges, the learning rate decreases gradually, which helps to prevent overshooting of the optimal solution, allowing fine-tuning and stable convergence. This effectively improved how the model generalised, leading to much better performance for unseen data.

**Slide 9: Training and Validation Performance**

Now we can take a deeper look into the model’s performance given the loss function and optimiser we have chosen, examining accuracy and loss trends over the epochs.

The graphs on the left shows that both the training and validation accuracy improved steadily over time, starting with a rapid increase initially as the model was able to learn key features quickly. However, the curve eventually plateaus as its ability to learn anything new from the dataset becomes much more difficult. What we can identify is that the model seems to generalise well without too much overfitting, shown by both accuracies being so similar.

The graph on the right shows the training and validation losses, with a decreasing loss indicating that as the training continues, the model is making better predictions. The slight fluctuations in the validation loss could be due to slight overfitting in the later stages, and the slight divergence between both lines show the importance of regularisation and data augmentation (Awan, 2024) to enhance generalisation.

We chose Stochastic Gradient Descent (SGD) with momentum, to be our optimiser for this project, which made iterative updates to the weights based on computed gradients, helping the model to converge effectively. We also reduced oscillations and improved stability, allowing the optimiser to maintain direction across updates, by using a momentum term set at 0.9. This helped for smoother convergence and stopped the model from getting stuck in local minima.

Monitoring these metrics has been essential in fine-tuning the model and making sure to balance both bias and variance, best we can.

**Slide 10: Regularisation Techniques Used**

To further improve the model’s ability to generalise towards unseen data, we used some regularisation techniques. One of these techniques was dropout, which we mentioned earlier, as it randomly deactivates neurons during training, to stop the model from relying on specific features of objects.

Furthermore, to prevent the model from just memorising the training data, and to encourage learning more meaningful features, we also applied L2 regularisation to the convolutional layers to penalise large weight values.

To test the effectiveness of regularisation, we compared validation accuracy when using and not using the techniques in the line graph here. Although the accuracy was initially lower without these techniques, the rate of loss did not improve, which meant early stopping was triggered and the model did not reach its highest potential accuracy, as the training stopped by the 14th epoch. Therefore, the regularisation techniques we used improved the model’s stability.

**Slide 11: Training Process**

Now we have discussed the details of creating our model, let’s look at the training process, outlined in our graphical process timeline. We planned to run the model for 100 epochs, with a batch size of 64. However, our use of early stopping caused the training to stop at around 50 epochs as the validation loss was not improving after this point. This helped to prevent overfitting and reduced unnecessary time and cost.

As discussed already, the accuracy and loss of our model was monitored for both the training and validation sets; the learning rate was adjusted dynamically based on performance, which is shown here. This ensured for optimal convergence.

Overall, the effectiveness of the training strategies employed were proven by the fact that the model’s ability to generalise was improving by each epoch.

**Slide 12: Model Evaluation Metrics**

Now that the training has been complete, we can evaluate the performance of the model through various metrics. Firstly, we look at the accuracy, which is the proportion of correctly classified images from the number of text samples. Our accuracy was calculated at 77.59%, indicating a decent performance, but also room to improve.

Accuracy is not the only important metric however as we also choose to evaluate our precision, recall, and F1-score metrics to gain a deeper insight into how effective the model is. The precision and recall evaluate the proportion of correctly identified positive objects, out of all of the predicted, and actual positives, respectively. The F1-score gives us a balance between precision and recall across different classes. These scores are also all relatively good, with some classes scoring better than others. Namely automobiles performing strongly and cats producing the poorest of results out of the classes.

**Slide 13: Confusion Matrix Analysis**

In order to understand how frequently each class was predicted correctly or incorrectly, we created a confusion matrix (C3.ai, n.d.). If we take a look at the y axis and follow the rows of frog and truck, we can see that they were both classified with high accuracy, whereas cats and dogs were frequently misclassified.

Now that we have identified where the misclassifications are occurring, we can try and identify any trends using the confusion matrix (C3.ai, n.d.). We can see that the two biggest misclassifications were when dogs were predicted as cats and cats were predicted as dogs, indicating that the model was unable to identify the subtle differences between the two animals, due to their similar features. Similarly to this, some automobiles were misclassified as trucks, also likely due to overlapping features such as wheels, logos, windows and colour patterns, despite their differences in size.

These observations have highlighted the need to improve the model’s ability to detect differences between visually similar classes; this can be done by enhancing our feature extraction techniques and making sure our data augmentation (Awan, 2024) methods are more targeted.

**Slide 14: Model Deployment Considerations**

Model deployment into real-world applications always comes with challenges which we need to address and overcome. Something which I noticed when taking part in this project is the strain on a device’s computational resources, especially edge devices, as they are limited. This is why it is so important to ensure the correct optimisation strategies are implemented, to ensure low latency and minimal memory footprint, while still maintaining accuracy. We could employ optimisation techniques such as quantisation and pruning to reduce the model’s size and inference time without having to severely compromise accuracy. This is extremely useful for deploying in resource constrained environments

There are still however, issues with cloud-based deployment, despite their scalability and ease of access, it comes with some latency and privacy concerns. Where immediate decision-making is vital, such as in the autonomous vehicles and surveillance industries, it may be more beneficial to use edge computing, as it provides real-time processing capabilities.

**Slide 15: Key Learnings and Challenges**

One of the most valuable insights that has emerged is how critical preprocessing and augmentation is in improving a model’s performance. Ensuring a model is able to generalise when looking at new data, ensures it can continue to predict correctly.

One of the difficulties we did face, was overfitting, which required fine-tuning of both regularisation and early stopping techniques as we did not want to deal with any excessive complexities. Also, a persistent challenge we faced was trying to enable the model to distinguish between classes with very similar features. For this to be avoided, we require more sophisticated methods of feature extraction in later iterations.

The final notable learning takeaway was how much the hyperparameter tuning affected the model’s performance. There was such a significant influence on the convergence and accuracy by just adjusting the learning rates, batch sizes and optimiser parameters alone.

**Slide 16: Future Work and Improvements**

For this project, I wanted to test myself by designing and training my own custom model from scratch. However, if we were to just be focussed on performance and not testing my own capabilities, the model performance could have been improved drastically by using pre-build models with convolutional architectures, such as ResNet or DenseNet, which are already proven to achieve high accuracy on similar classification tasks. Additionally, we could have incorporated transfer learning by using pre-trained models to improve feature extraction and enable the model to successfully distinguish between object classes with similar characteristics.

When assessing our model, there are still areas which need improving, one being hyperparameter tuning. Further exploration into this area could have improved our results; finding the perfect learning rates, batch sizes and parameters should be a main focus in the future, but on this occasion my focus was wider spread on experimenting with every aspect I could investigate.

Finally, future work could involve deploying the model in real-world scenarios to assess its practicality and find potential areas for refinement based on real-time performance feedback.

**Slide 17: Conclusion**

In conclusion, we have explored the effectiveness of convolutional neural networks in performing object recognition using the CIFAR-10 dataset (Krizhevsky, 2009). We created a model which achieved a commendable 77.59% accuracy and investigated how we can also make improvements in the future, not forgetting to highlight the many strengths the model possesses.

As discussed already, there are several key takeaways from this project: the impact of changes to the architecture, regularisation techniques and the importance of generalisation. Moving forward, further refinements are definitely necessary to improve the model’s performance and adaptability to real-world applications.

Thank you for your attention and I look forward to any feedback you may have.

## References

Krizhevsky, A. and Hinton, G., 2009. Learning multiple layers of features from tiny images. [online] Available at: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf> [Accessed 6 December 2024].

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp.1097-1105. [online] Available at: <https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf> [Accessed 2 January 2025].

He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.770-778. [online] Available at: <https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf> [Accessed 8 January 2025].

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V. and Rabinovich, A., 2015. Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp.1-9. [online] Available at: <https://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Szegedy_Going_Deeper_With_2015_CVPR_paper.pdf> [Accessed 28 December 2024].

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations. [online] Available at: <https://arxiv.org/pdf/1409.1556.pdf> [Accessed 20 January 2025].

Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A. and Bengio, Y., 2013. Maxout networks. In: Proceedings of the 30th International Conference on Machine Learning, pp.1319-1327. [online] Available at: <http://proceedings.mlr.press/v28/goodfellow13.pdf> [Accessed 3 January 2025].

Springenberg, J.T., Dosovitskiy, A., Brox, T. and Riedmiller, M., 2015. Striving for simplicity: The all convolutional net. In: International Conference on Learning Representations (Workshop Track). [online] Available at: <https://arxiv.org/pdf/1412.6806.pdf> [Accessed 5 January 2025].

Ioffe, S. and Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning, pp.448-456. [online] Available at: <http://proceedings.mlr.press/v37/ioffe15.pdf> [Accessed 18 January 2025].

Krizhevsky, A., 2009. CIFAR-10 and CIFAR-100 datasets. [online] Available at: <https://www.cs.toronto.edu/~kriz/cifar.html#:~:text=CIFAR-10%20python%20version> [Accessed 27 January 2025].

C3.ai, n.d. Evaluating Model Performance. [online] Available at: <https://c3.ai/introduction-what-is-machine-learning/evaluating-model-performance/> [Accessed 16 December 2024].

Nayda, Y., 2024. 10+ Applications of Object Detection Technology in Various Industries. [online] SmartTek Solutions. Available at: <https://smarttek.solutions/blog/object-detection-technology/> [Accessed 9 January 2025].

Baheti, P., 2021. Activation Functions in Neural Networks [12 Types & Use Cases]. [online] V7 Labs. Available at: <https://www.v7labs.com/blog/neural-networks-activation-functions> [Accessed 20 January 2025].

Gómez, R., 2018. Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names. [online] Raúl Gómez Blog. Available at: <https://gombru.github.io/2018/05/23/cross_entropy_loss/> [Accessed 12 January 2025].

Awan, A.A., 2024. A Complete Guide to Data Augmentation. [online] DataCamp. Available at: <https://www.datacamp.com/tutorial/complete-guide-data-augmentation> [Accessed 4 January 2025].

Hugging Face, n.d. Model Deployment Considerations. [online] Available at: <https://huggingface.co/learn/computer-vision-course/unit9/model_deployment> [Accessed 19 January 2025].

IBM, 2021. What is overfitting? [online] Available at: <https://www.ibm.com/think/topics/overfitting> [Accessed 19 December 2024]

Zhang, C. et al., 2021. ResNet or DenseNet? Introducing Dense Shortcuts to ResNet. [online] Korea Advanced Institute of Science and Technology. Available at: <https://openaccess.thecvf.com/content/WACV2021/papers/Zhang_ResNet_or_DenseNet_Introducing_Dense_Shortcuts_to_ResNet_WACV_2021_paper.pdf> [Accessed 23 January 2025].