Neural Network Models for Object Recognition

Individual presentation

Transcript

Slide 1:

Good afternoon, everyone. Today, I will be talking about neural networks, which are a fundamental component of deep learning, a subset of artificial intelligence.

Slide 2: Introduction to Neural Networks

Let’s begin with an introduction to neural networks. Neural networks are designed to mimic the human brain's ability to process and learn from data. Its structure enables neural networks to excel in tasks such as image recognition. According to LeCun et al. (2015), the ability of neural networks to learn from large datasets is what makes them particularly powerful.

Slide 3: Introduction to Training

Now, let’s talk about the training process. Training a neural network involves an iterative process known as an epoch. During each epoch, the training data is passed through the network, and the loss—a measure of the difference between the predicted and actual outputs—is calculated. As Goodfellow et al. (2016) explain, the loss function is critical in guiding this optimisation process. For example, Mean Squared Error (MSE) is commonly used for regression tasks, while Cross-Entropy Loss is popular for classification tasks.

Slide 4: Key Metrics for Performance

Moving on to key metrics for performance. During training, two key metrics are used to evaluate the network's performance: training loss and validation loss. A well-trained model will have low training and validation losses, indicating it has learned effectively without overfitting to the training data (GeeksforGeeks, 2024).

The architecture of a neural network also plays a crucial role in its performance. For instance, activation functions like ReLU, which are commonly used in hidden layers, help the model learn complex, non-linear relationships. For multi-class classification tasks, softmax is often used in the output layer to assign probabilities to each class. As Ali (2024) highlights, selecting the right activation functions is essential for optimising model performance.

Slide 5: Introduction to Convolutional Neural Networks (CNNs)

One popular type of neural network used for image-related tasks is the Convolutional Neural Network, or CNN. CNNs are designed to automatically detect important features in images, such as edges, textures, and shapes, by applying convolutional filters in multiple layers. This architecture allows CNNs to excel in image classification tasks, like those involving the CIFAR-10 dataset (Spiceworks, 2025).

Slide 6: The Dataset

The CIFAR-10 dataset is a widely used collection of labelled images designed for machine learning and computer vision tasks. It was created by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The dataset consists of 60,000 colour images, each with a resolution of 32x32 pixels. These are categorised into 10 distinct classes, with 6,000 images per class. This results in 50,000 training images and 10,000 test images. The dataset is divided into five training batches and one test batch, each containing 10,000 images. The test batch includes 1,000 randomly selected images from each class, while the training batches contain the remaining images in random order (Krizhevsky, 2009).

Slide 7: Objective

Developing a neural network model using the CIFAR-10 dataset for object recognition.

Slide 8: Methodology

For the image classification task, I implemented a Convolutional Neural Network (CNN) using TensorFlow and Keras. Below, I discuss the decisions made, the rationale behind them, and the insights I gained from external resources. I also include a critical evaluation of the choices and their impact on the model’s performance.

Slide 9: Validation Set Partitioning

The provided code snippet demonstrates validation set partitioning using the train_test_split function from scikit-learn. It splits the original training data (X_train, containing features, and y_train, containing corresponding labels) into two subsets: a training set for model training and a validation set for evaluating performance and tuning hyperparameters. The test_size=0.1 argument allocates 10% of the data to the validation set. Setting random_state=0 ensures consistent splits across runs. After this split, X_train and y_train contain the reduced training data, while X_valid and y_valid hold the features and labels for the validation set. This technique is crucial in machine learning to prevent overfitting and optimise model hyperparameters.

Slide 10: Data Processing

For normalisation, I scaled the pixel values to a range of [0, 1] to standardise the data, ensuring that the model could learn efficiently. This preprocessing step is standard in image classification tasks and helps the model converge faster (Waiss, 2023; Chng, 2022).

To combat overfitting and increase the generalisation ability of the model, I applied data augmentation techniques such as random horizontal flipping, rotations, and shifts. While these methods introduced variability, they also helped the model avoid memorising the training data and improved its performance on unseen data. However, I was cautious not to overdo augmentation, as excessive transformations can lead to unrealistic training examples that could negatively impact model performance.

Slide 11: Model Architecture

I experimented with different filter sizes for the convolutional layers. Initially, I used 8 filters, which resulted in overfitting. The model performed well on training data but struggled to generalise to the test set. This issue is common when the model is too complex relative to the dataset size, as it learns to memorise rather than generalise. I reduced the number of filters to 3, which improved the model's generalisation. However, this choice still raised the question of whether a deeper or more complex architecture could have been beneficial. While reducing the complexity of the model helped with overfitting, it also limited the model's capacity to capture more complex patterns in the data. A trade-off between complexity and generalisation must always be considered (Hopswork, 2021).

I used max-pooling layers after each convolutional layer to reduce the spatial dimensions of the feature maps. Pooling layers help in reducing the computational load and prevent overfitting by making the model more invariant to small translations in the image. However, max-pooling can sometimes discard valuable spatial information. Alternative pooling strategies like average pooling or global pooling might have preserved more spatial features but would have increased the model's complexity and training time.

To further combat overfitting, I added dropout layers after the dense layers. Dropout randomly sets a fraction of input units to zero during training, which forces the model to learn redundant representations and prevents it from relying too heavily on any single feature. However, dropout is a hyperparameter that needs fine-tuning. If set too high, it can hinder learning by removing too much information. The dropout rate I chose was balanced, but further experimentation could refine this choice.

Slide 12: Learning Rate

I experimented with a range of learning rates, from 0.1 to 0.0001. Initially, I found that a learning rate of 0.1 was too high, causing the model to diverge, while 0.0001 was too low, resulting in slow convergence. The learning rate of 0.001 struck a balance, allowing the model to converge steadily without oscillating.

I later went back to using the Adam optimiser, which did aid in my convergence. One of the limitations of this approach is that the learning rate was chosen manually based on trial and error. Automated hyperparameter optimisation methods, such as cyclical learning rates could have been used to more systematically explore the optimal learning rate and other hyperparameters, potentially leading to better performance.

Slide 13: Activation Function

I used the ReLU (Rectified Linear Unit) activation function for the convolutional and dense layers. ReLU is widely used in CNNs because it helps mitigate the vanishing gradient problem and allows the model to learn non-linear relationships more efficiently. However, ReLU can sometimes suffer from the "dying ReLU" problem, where neurons stop learning entirely due to negative inputs. To address this, I could have experimented with variations of ReLU, such as Leaky ReLU or Parametric ReLU, which allow small negative values to pass through, potentially improving performance (Ali, 2024).

Slide 14: Loss Function and Optimiser

I used the categorical cross-entropy loss function, which is appropriate for multi-class classification tasks. The Adam optimiser was chosen due to its adaptive learning rate, which helps in faster convergence and better performance.

However, while Adam generally performs well, it might not always be the best choice. Other optimisers like SGD with momentum or RMSprop could have been tested to see if they offered better performance on this specific task. The choice of loss function and optimiser is crucial for model performance. Experimenting with other loss functions, such as focal loss for imbalanced classes, might have improved the model’s robustness, especially in more complex or imbalanced datasets.

Slide 15: Evaluation Metrics

I did not choose accuracy as the primary evaluation metric, despite its common use in classification tasks. While accuracy monitors the model's performance, I found that loss and validation loss were more reliable indicators of model performance, particularly when trying to avoid overfitting.

In the initial attempts, the model showed great accuracy (84%), but the validation loss was higher, suggesting that the model was overfitting to the training data. This is a common issue in deep learning, where a model can perform well on training data but fail to generalise to unseen data (GeeksforGeeks, 2021).

While focusing on validation loss was a reasonable decision, I could also put more emphasis on additional metrics like precision, recall, or F1 score to provide a more comprehensive evaluation of the model, especially in cases where class imbalance is a concern. Although I performed these evaluations, I did not focus on them to the extent that I focused on validation loss. In retrospect, I think it would have made fine-tuning the model easier. These metrics would have given a clearer picture of how well the model is performing across different classes, particularly in real-world scenarios where class distributions are often skewed (GeeksforGeeks, 2024).

Slide 16: Inspiration and Resources

I drew inspiration from the activity provided in Unit 9, which helped me understand the basics of CNNs and image classification. However, I also sought additional insights from a Kaggle notebook by Farzad Nekouei titled "CIFAR-10 Image Classification with CNN" (Nekouei, 2023).

This resource provided a detailed explanation of the metrics and methods used, along with reasoning behind the choices. It also gave me ideas on how to improve my model’s performance, such as the use of specific data augmentation techniques and a deeper understanding of hyperparameter tuning. However, while the Kaggle notebook was informative, I realised that blindly following external sources can sometimes lead to overfitting to the specific methods used in those sources, which happened when I first tested 8 convolutional layers.

It’s crucial to adapt ideas to the specific problem at hand and experiment with different approaches to find the optimal solution. I also found Stack Overflow to be a great resource for understanding errors in my coding and gaining different perspectives to solve errors.

Slide 17: Findings Introduction

The next few slides will dive into my findings. These include model performance, evaluation metrics, and insights from classification reports and confusion matrices.

Slide 18: Model Performance

For model accuracy evolution, the training and validation accuracy lines generally show an upward trend, indicating that the model's accuracy improved over time. However, the blue line, representing training accuracy, eventually becomes higher than the orange line, which represents validation accuracy. This is a common phenomenon called overfitting, where the model learns the training data too well, leading to poor generalisation on unseen data.

For gradient descent, both lines generally show a downward trend, indicating that the model's loss decreased over time. This is a good sign, as it means the model was learning and improving. However, the training loss was consistently lower than the validation loss, again indicating overfitting.

The test accuracy represents the percentage of correct predictions the model made on the validation dataset. The test loss represents the average error of the model's predictions on the validation dataset. The model achieved a test accuracy of 83.42% and a test loss of 0.5210677981376648. This suggests that the model performed reasonably well on unseen data.

Slide 19: Classification Report

The classification report provides a summary of the model's performance on a classification task.

The model achieved an overall accuracy of 83%. The macro average and weighted average scores are also around 83%, indicating consistent performance across classes. Some classes have higher precision and recall than others. For example, class 1 has very high precision and recall, while class 3 has lower scores.

Slide 20: Confusion Matrix

The confusion matrix is a visual representation of the performance of a classification model. In this specific matrix, the model performed well on most classes, with high values along the diagonal. However, there were some misclassifications between certain classes, such as "cat" and "dog". Some classes had more misclassifications than others.

Overall, this matrix gives a good overview of the model's performance and helps identify areas for improvement.

Slide 21: Final Prediction

The array of numbers below the predicted class represents the probabilities assigned by the model to each of the possible classes. In this case, there are 10 possible classes, as indicated by the 10 values in the array.

The highest probability value (0.46335021) corresponds to the "truck" class, which confirms the model's prediction.

In summary, the model successfully identified the image as depicting a truck with a high degree of confidence.

Slide 22: Challenges and Insights

A significant challenge was balancing model complexity and generalisation, as overly complex architectures led to overfitting, while simpler models struggled to capture intricate patterns in the data. Hyperparameter tuning, such as selecting the optimal learning rate and dropout rate, proved to be a time-intensive process requiring iterative adjustments.

Additionally, the uneven distribution of class performance, as revealed by the classification report and confusion matrix, highlighted the difficulty of achieving consistent accuracy across all classes.

These challenges offered insights into the importance of preprocessing techniques, such as data augmentation, and the need for advanced evaluation metrics to gain a comprehensive understanding of model performance. The iterative nature of experimentation demonstrates the value of systematic approaches, such as automated hyperparameter optimisation, and the potential for deeper architectures or transfer learning to enhance performance.

To gain a deeper understanding of the model's performance and identify areas for improvement, I would test a set of dog and cat images. By analysing the misclassifications of these classes we can assess how these errors impact the overall accuracy of the results.

To enhance the performance of this model, utilising a larger dataset could be beneficial. A larger dataset would allow for the incorporation of deeper convolutional neural networks with increased numbers of layers. This deeper architecture would enable the model to extract more intricate features from the images, potentially leading to more accurate predictions.

Slide 23: Conclusion

The development of a Convolutional Neural Network (CNN) for object recognition using the CIFAR-10 dataset achieved a test accuracy of 83.42%. Key factors contributing to the success included effective preprocessing, balanced architecture, and regularisation methods that reduced overfitting. Hyperparameter tuning, such as adjusting the learning rate and dropout rate, was crucial for performance. However, challenges like overfitting to specific classes and manual hyperparameter tuning suggest areas for improvement. Future work could focus on automated hyperparameter optimisation, deeper architectures, and advanced data augmentation to further enhance accuracy and eventually leverage deep learning to solve complex real-world problems.

Slide 24: Ethical and Real-world problems

Machine learning models, while incredibly useful, come with ethical challenges, especially when applied to real-world situations. Take, for example, this model's misclassification of cats and dogs. While this might seem harmless, it highlights a bigger issue. If a model makes a mistake and misclassifies a cat as a dog, it might not matter much in this case. However, in more critical areas like healthcare or self-driving cars, the consequences of a wrong decision could be much more serious. For example, a misdiagnosis in healthcare could lead to the wrong treatment. These situations show just how important it is to make sure the data used to train these models is fair and representative. If the model is trained on biased or incomplete data, it can make decisions that unfairly affect certain people. So, when using machine learning in real-world applications, it's essential to be aware of the ethical risks and ensure that the technology is used responsibly, especially in situations where people's lives are at stake.

Slide 25: References

Here are all the references I used. My sources range from research papers to web articles, blogs, and forum posts made by my peers.

Slide 26: Closing

We have reached the end of my presentation. Thank you for listening.