Assessment 1 – Neural Network Models for Object Recognition

Student ID: 12696139

Transcript File

Assignment (Slide 1)

I am Murthy Kanuri, a student in the Master of Science in Artificial Intelligence programme. This assignment is part of the Machine Learning module and involves developing a Neural Network model for object recognition using the CIFAR-10 image dataset.

Introduction (Slide 2)

Object recognition is one of the great capabilities of artificial intelligence. Systems interpret the appearance or presence of objects from images and video streams, and this process is the foundation for many real-world scenarios for example:

- Self-driving cars like Tesla use object recognition to monitor traffic lights, pedestrians, and other traffic obstacles and respond correctly to safe navigation with appropriate decision-making.
- Facial recognition is used to secure devices and locations. It unlocks smartphones, which has become convenient for many people worldwide.
- In security and monitoring, object recognition can detect suspicious activities, unauthorised access, or the presence of specific individuals or items, improving public safety and operational efficiency.

It is indisputable that machine learning algorithms, most notably deep learning, have been at the forefront of the evolution of object recognition. They allow systems to sift through vast amounts of visual data with a very high level of accuracy.

CIFAR-10 Dataset (Slide 3)

This presentation aims to implement and evaluate the performance of Convolutional Neural Networks (CNNs) on the CIFAR-10 dataset, a widely used benchmark in image classification tasks.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton introduced the CIFAR-10 dataset in 2009.

The CIFAR-10 dataset consists of 60,000 colour images with dimensions of 32x32 pixels, divided into 10 distinct categories, each containing 6,000 images. Each image has 3 RGB colour channels.

The CIFAR-10 dataset is split into a training set with 50,000 images and a test set with 10,000 images.The dataset has 10 class categories.

CIFAR-10 Dataset (Slide 4)

This image provides a visual overview of the CIFAR-10 dataset, featuring sample images from each of the 10 unique classes. The categories are displayed on the left, with their respective images in rows.

Each row showcases the differences within the class, highlighting the variety of objects and scenes included in the dataset. These images emphasise image classification challenges, such as differences in lighting, angles, and object appearances.

Required Libraries (Slide 5)

In this part of the project, we import several essential libraries and modules to build and evaluate the Convolutional Neural Network (CNN) model for classifying images in the CIFAR-10 dataset using Google Colab.

A list of libraries, along with their usage descriptions, is provided below.
Python's extensive support and ease of use make it the ideal choice for deep learning projects, such as image classification with CIFAR-10.

Partitioning the validation set and data insights (Slide 6)

- Scikit-learn's "train_test_split" function divides the data into three sets: 80% is designated for training, while the remaining 20% is split evenly between validation and testing.
- This hierarchical breakdown ensures that the dataset is partitioned correctly, facilitating the training, validation, and testing of machine learning models.
- The stratify parameter ensures that the class distributions remain consistent across splits.
- Class distributions are checked to confirm the integrity of the splits.

Training, validation and testing set details (Slide 7)

This slide provides an overview of the training, validation, and testing sets, which are subsets of a dataset used in machine learning to train, validate, and evaluate models. Each subset serves a specific purpose, ensuring the model generalises well to unseen data.
Typically, 60-80% of the dataset is allocated for training, 10-20% for validation, and 10-20% for testing.

Visualising Class Distributions (Slide 8)

- This slide visualises the class distributions of the CIFAR-10 dataset across the training, validation, and testing sets, ensuring balanced representation among the 10 classes.
- **The Training Set on the left** shows the distribution of classes in the training set, with approximately 4,000 samples per class.
- **The Validation Set in the Middle d**isplays the validation set's class distribution, with around 1,000 samples per class.
- **The Testing Set on the right** depicts the testing set's distribution, also containing about 1,000 samples per class.
- **Importance of validation Set (Slide 9)**
- This slide highlights the importance of using a validation set. In summary, the validation set plays a crucial role in model evaluation, hyperparameter tuning, preventing overfitting, enabling early stopping, and assisting in model selection.

Data Preparation - Normalisation (slide 10)

- In the CIFAR-10 dataset, pixel intensity values range from 0 to 255. To enhance the model's performance, these values are normalised to a [0,1] range by dividing by 255, as shown in the equation.
- Normalisation improves model efficiency by preventing large pixel values from dominating learning and ensures CNN detects patterns better.
- On the left side of the slide, it shows original and normalised automobile images.
- The right side of the slide shows Histograms showing the pixel intensity distribution before (0-255) and after normalisation (0-1).

Normalisation (Slide 11)

- On the left side of the slide, we see the pixel values of the first image in the dataset before normalisation.
- In the middle, we have the data preprocessing step, dividing the pixel values by 255.
- On the right side of the slide, we observe the pixel values after normalisation.

One-Hot encoding (Slide 12)

- This slide covers One-Hot Encoding, a crucial step in preparing CIFAR-10 labels for neural networks.

- The code snippet demonstrates One-Hot Encoding using the to_categorical function and printed dataset shapes to confirm successful encoding.

- In summary, One-Hot Encoding improves model interpretability and ensures compatibility with loss functions like categorical cross-entropy.

Artificial Neural Network (ANN) (Slide 13)

- ANN works in a similar format to the human brain. It comprises interconnected layers of neutrons (nodes), where each neutron processes input data and passes the output to the next layer.
- As the network's first layer, the input layer handles raw data intake. Hidden layers extract and transform features using weighted connections and activation functions, introducing non-linearity. The output layer provides final predictions based on processed data from the hidden layers.

Convolution Neural Network (CNN) (Slide 14)

In the previous slide, we covered Artificial Neural Networks (ANNs). While ANNs work well for structured data, CNNs are better for image-based datasets like CIFAR-10 due to their ability to capture spatial patterns efficiently. CNNs also enhance computational efficiency through weight sharing and reduce overfitting with pooling and dropout layers.

The diagram illustrates the CNN architecture, emphasising two key steps: Feature Learning and Classification.

- Input Layer: The process starts with an input image (e.g., a car), which enters the network for analysis.
- Feature Learning:
    - Convolutional layers apply filters to detect features such as edges and shapes, producing feature maps.
    - Activation functions (e.g., ReLU) add non-linearity while pooling layers reduce spatial dimensions.
    - This sequence of convolution, activation, and pooling is repeated to extract complex features.
- Classification:
    - The extracted features are flattened into a 1D array and processed by fully connected layers.
    - A softmax function in the output layer classifies the image into predefined categories like cars or trucks

Chosen Activation Function- ReLU & Softmax (Slide 15)

- This slide provides an overview of the chosen activation functions for our model—ReLU and Softmax. ReLU is applied to the hidden layers, introducing non-linearity and improving efficiency by allowing sparse activations. It is particularly effective in preventing

the vanishing gradient problem, which commonly affects functions like Sigmoid and Tanh.
- On the other hand, Softmax is used in the output layer to convert the model's raw outputs into probabilities, ensuring that predictions are interpretable for classification tasks.
- The table summarises their mathematical expressions, range, and key characteristics, highlighting why these functions are well-suited for our model.

Chosen Activation Function- ReLU & Softmax (Slide 16)

- This slide presents visual representations of both activation functions.
- The ReLU graph clearly demonstrates that it outputs zero for negative values while allowing positive values to pass through.
- The Softmax graph, on the other hand, displays its characteristic S-shaped curve, which maps raw outputs to probability distributions.
- In short, these graphs reinforce visually the theoretical aspects from the previous slide of how each of these functions works in practice.

Implemented Loss Function (Slide 17)

This slide explains the Categorical Cross-Entropy (CCE) loss function, commonly used in classification tasks. It measures how well the predicted probability distribution matches the true labels, penalising incorrect predictions to improve accuracy.

The formula shown calculates the loss by summing the true class labels multiplied by the logarithm of predicted probabilities.

CCE works best with the softmax activation function, which converts model outputs into probabilities, ensuring they sum to one—making it ideal for multi-class classification.

Build the Model (Slide 18)

This slide presents our CNN model for the CIFAR-10 dataset, built using Keras Sequential API.
- The input layer processes 32x32 RGB images.
- The model consists of **three convolutional blocks**, each with:
  - Two convolutional layers (increasing filters from 32 to 128),
  - Batch normalisation for stable learning,
  - Max pooling to reduce spatial dimensions,
  - Dropout to prevent overfitting.
- After feature extraction, the model flattens the data and passes it through **fully connected layers.**

- The model contains two dense layers with 256 and 128 neurons, each with ReLU activation.
- Dropout layers enable regularisation, and the softmax output layer classifies the images of 10.

The architecture learns features gradually and effectively, increasing the model's capability to classify images in the CIFAR-10 dataset.

Model Summary (Slide 19)

- This slide presents the model summary, detailing the layer types, output shapes, and total parameters.
- The model starts with convolutional layers that process 32x32 RGB images, followed by batch normalisation for stability and max pooling to reduce spatial dimensions.
- The convolutional blocks progressively increase filter sizes from 32 to 128, allowing the model to capture deeper features, with parameters growing accordingly.
- Flattening converts feature maps into a 1D vector, preparing data for fully connected layers.
- The dense layers contain 256 and 128 neurons, followed by dropout layers to prevent overfitting. The final softmax layer classifies images into 10 categories.
- The model comprises **847,530 parameters**, with **846,634 trainable** and **896 non-trainable**, primarily from batch normalisation layers.
- This summary highlights how the model balances complexity with efficiency to achieve effective learning.

Compile the Model (Slide 20)

- After building the model, we compile it using the Adam optimiser and categorical cross-entropy loss function.
- Adam optimiser dynamically adjusts learning rates for efficient training.
- The categorical cross-entropy loss function is applied in multi-class classification tasks
- The accuracy metric helps track performance during training.
- Compiling with Adam ensures faster convergence and effective learning for the CIFAR-10 dataset.

Train the Model (Slide 21)

- The **model.fit** function is responsible for training the model with both training and validation data using key parameters such as:
- **Epochs (50):** Specifies the number of complete passes through the dataset.
- **Batch size (64):** Defines how many samples are processed at a time.
- **Early stopping:** Tracks validation loss and halts training if no improvement is detected for five consecutive epochs, restoring the optimal weights.
- These configurations enhance training efficiency and help prevent overfitting, leading to improved generalisation.

Train Log / Training Progress Output (Slide 22)

- This slide presents the training log, showcasing key performance metrics over multiple epochs.
- The model was trained for up to 50 epochs, with early stopping to prevent overfitting when validation loss stops improving.
- Metrics such as accuracy and loss track training progress showing a steady improvement in performance.
- Validation accuracy and loss help assess generalisation to unseen data, with fluctuations early on that stabilise over time.
- By the 22nd epoch, the model achieved 85.95% training accuracy and 81.76% validation accuracy, indicating good generalisation.
- This log helps in monitoring training progress and guiding potential adjustments to improve performance.
-

Evaluate the Model (Slide 23)

- Upon training completion, the model is evaluated on the test set to assess its generalisation performance.
- Accuracy: 83.55% indicates strong performance in classification on test data.
- The test loss is **0.5467**, reflecting the model's prediction error.
- These results demonstrate good generalisation, though further improvements could be made through techniques like data augmentation or hyperparameter tuning.

Confusion Matrix & Classification Report (Slide 24)

This slide presents the confusion matrix and classification report, which evaluate the model's performance.

- **Confusion Matrix:**
  - Visualises true vs. predicted labels.

    - Diagonal values indicate correct classifications, while off-diagonal values show misclassifications.
    - The model performs well on classes like automobiles and ships but shows confusion between similar classes such as dogs and deer.
- **Classification Report:**
    - Provides precision, recall, and F1-score for each class.
    - It reached 82% accuracy, indicating a balanced performance for all categories.
    - These insights point to areas for possible improvement, such as fine-tuning or data augmentation.

Calculate Metrics for Each Class (Slide 25)

This slide provides some performance metrics on a class-by-class basis, such as true positives, false positives, false negatives, and true negatives.

- True positives indicate correctly classified samples, and false positives indicate misclassified samples.
- Analysing these values helps show which classes perform well and which need improvement.

Evaluating these metrics is very important for tuning the model. High false positives or false negatives suggest areas for enhancement through feature engineering or model retraining.

Model Accuracy & Model Loss (Slide 26)

- On this slide, the accuracy graphs the loss graph, respectively, present the model's learning progress at 25 epochs.
- The accuracy plot illustrates a steady increase in the training accuracy which, for the most part, exceeds 85%.
- Validation accuracy also generally increases, being of the order of magnitude of the training curve, implying a relatively small amount of overfitting.
- Nevertheless, a narrow residual distance between the two suggests that the generalisation of the model can be further improved by adding the gap.
- In the loss graph, both training and validation losses decrease with time, demonstrating learning. At a specific stage, the evaluation loss begins to stabilise, which suggests, overfitting phenomenon.

Visualise Some Predictions (Slide 27)

This slide visually evaluates the model's predictions on the CIFAR-10 test set.

- The displayed images, along with their true and predicted labels, provide a visual way to evaluate the model's performance.
- Observing both correct and incorrect classifications highlights areas that need improvement.
- Visualising these results helps identify misclassification patterns and guides the necessary adjustment and optimisation of the model.
- Analysing these insights enables refining preprocessing techniques and improving future iterations of the mode

Performance Insights from Sample Predictions (Slide 28)

- This slide presents the classification performance of the model by showing the predicted instances in different classes. Although at its peak in discrete object recognition (ships, aeroplanes, frogs) the model is less successful in handling visually alike categories (dogs or deer), for which there is occasional mistaking.
- These findings point to the fact that the model captures discernible patterns well but may need a respective adaptation to differentiate between highly related classes. The scope for improvement may include either continued sophistication in feature extraction techniques or the enhancement of the training set to more clearly separate classes.

Insights into Neural Network Design Strategy (Slide 29)

- In this assignment, the Convolutional Neural Network (CNN) model was successfully utilised for the image classification task on the CIFAR-10 dataset.
- We followed a structured approach, which included preprocessing the data using normalisation and one-hot encoding, designing the CNN architecture, training the model with the Adam optimiser, applying activation and loss functions, implementing the early stopping technique, and evaluating the model's performance.
- The CNN proved to be highly accurate on the test data, showing the CNN's capabilities of image classification in CIFAR-10.

Reflections on Learnings (Slide 30)

- I developed a solid understanding of neural network models and their applications in various real-world scenarios, such as object recognition.
- Handling datasets has given me valuable insights into the significance of preprocessing techniques.
- Working with TensorFlow and Keras has given me practical, hands-on experience in efficiently building, training, and fine-tuning models.
- Building the CNN model gave me valuable insights into important performance metrics: accuracy, precision, recall, and loss.

- I also learned that using a GPU in Google Colab instead of a CPU significantly enhanced my model's training speed and efficiency.
- This work emphasised the crucial need to address ethical concerns, such as minimising dataset bias and ensuring fairness in AI models, to achieve reliable and unbiased outcomes.
- Discussing ideas with my peers allowed me to learn complex concepts more effectively and gain new perspectives.
- Staying updated with industry trends through research papers and online courses has helped me stay informed about deep learning advancements and apply them effectively in my projects. I am planning to continue my further learning by participating in online seminars and discussions.