**INTRO PAGE**This assignment will be used for the partial fulfillment of requirements of Module 3 Machine Learning that started October 2024as part of the PGD Certificate in Artificial Intelligence at University of Essex Online.

**Executive Summary:**

* The assignment aimed to train a Neural Network using the dataset of the Canadian Institute For Advanced Research-10, in short CIFAR 10. The task was to evaluate the performance of the dataset, and use Python for data mining. Key aspects included data preparation, model architecture, model training and design strategies. Reflections on the process, challenges and learnings were also essential components. A Convolutional Neural Network (CNN) was chosen as the preferred deep learning model since it is perfect for structured data like images. Since the dataset CIFAR 10 consists of images, this model is very good in the context of object recognition.
* The problem statement revolves around exploring whether a Convolutional Neural Network (CNN) can effectively be trained for object recognition using the CIFAR-10 dataset, and if so, to assess its performance and accuracy in identifying objects across ten distinct categories.
* Key outcomes included a 55.22% accuracy and a loss of 1.24, showing a good result that could have naturally been even better. The model struggled with underrepresented classes like "cat" and "bird" and similar pairs such as airplanes vs. ships (0 vs. 8) and cats vs. frogs (3 vs. 6).
* Working on model refinement throughout the course of an entire week improved stability, and programming skills, but there is still room for improvement.

**Agenda**This presentation is structured into 4 parts, starting with the Introduction, continuing with the workflow of the model development and afterwards going into model optimization and validation. Learnings and other insights finalize this presentation.

**Introduction: Training CNNs for Object Recognition with CIFAR-10**  
Artificial Neural Networks (ANNs) are a subset of AI systems, often described as functioning in a manner similar to the human brain. They are widely applied in various domains, with speech and image recognition being among the most prominent use cases, as described by Chi Leung Hui (Chi Leung Hui, P., 2023).

* According to Al-Malah, Neural networks are popular due to their capacity to learn from data and make predictions or informed decisions. Subsets of image recognition include facial recognition, object detection, and image classification. Neural Networks are being applied in these areas to assist in interpreting visual data in security systems, social media, and autonomous vehicles (Al-Malah, K.I.M. 2024).
* Al-Malah continues to say that Convolutional Neural Networks (CNNs) are a type of deep learning model optimized for handling structured data like images, transforming inputs through weighted layers into 3-D outputs known as activations. They are versatile enough to be used in areas such as text, audio and signal processing, with their structure customized for specific tasks by varying the Layer configurations and quantities (Al-Malah, K.I.M. 2024).
* According to Aceves-Fernandez, a loss function is crucial for evaluating the performance of a neural network and CNN training is focused on minimizing exactly this loss (Aceves-Fernandez, M. A. 2020).
* The objective of this assignment, finally, is training a CNN for object recognition using the CIFAR-10 dataset and on this slide we see a sample of labeled images from this very same dataset. The assignment covered dataset preparation, model design, training, evaluation, and insights.

**Understanding the CIFAR-10 Dataset and the split methodology**

* Let’s address why CIFAR-10 was chosen for this project. As Mattmann highlights, it is a widely used benchmark dataset for image classification tasks. It contains 60,000 images, each measuring 32x32 pixels, divided into 10 categories such as airplane, automobile, bird, cat, and others. Its manageable size makes it an ideal choice for this assignment, because it has the right balance between being complex and yet feasible for the use case of developing and testing visual recognition models.
* The common datasplit of 80/10/10 was chosen, meaning 80% for training, 10% for validation, and 10% for the test set, resulting in 48,000 training and 6,000 validation and test images each.

Witten states that a Validation Set is Used during training to adjust hyperparameters and optimize model complexity, ensuring the model does not overfit or underfit while adjusting its architecture or parameters.

He continues to say that the Test Set is Used after training to evaluate the final model's performance on unseen data, ensuring unbiased metrics and confirming generalization.

**Data Exploration and Neural Network Architecture**

We continue with the second part of the presentation, which focuses on the Model Development Workflow. To begin, the CIFAR-10 dataset was prepared by loading the data and normalizing pixel values to a range between 0 and 1, making it easier for the model to process. The dataset was divided into training, validation, and test sets in an 80/10/10 ratio, ensuring sufficient data for each stage of training and evaluation. The labels were one-hot encoded to align with the model's requirements. To better understand the dataset, a sample image with its label was displayed, along with a grid view of training images, allowing us to analyze the diversity of classes. Data analysis also included verifying the shapes of the data and examining class distributions to confirm that the dataset was ready for modeling.

The neural network was designed with three convolutional layers, each using Rectified Linear Unit (ReLU) activation, batch normalization, max-pooling, and dropout. These layers worked together to extract features effectively while minimizing overfitting. A dense layer with 256 units was added next, also using ReLU activation and dropout, to further refine learning and ensure generalization. The reason the Rectified Linear Unit was used throughout the convolutional and fully connected layers because it helps the model train efficiently by preventing issues like weakening gradient signals. The final layer employed softmax activation to convert the network’s outputs into probabilities, predicting the likelihood of each input belonging to one of the 10 categories.

The model was trained using the Adam optimizer, which is known for its adaptive learning capabilities, and the categorical cross-entropy loss function, which is well-suited for multi-class classification tasks. During training, accuracy tracking was used to monitor the model’s performance. A utility tool was also implemented to visualize images and their labels, offering deeper insights into the model’s structure, important components, and overall complexity.

The bar chart shown here illustrates how computational demands are distributed across the layers. Most of the workload is handled by the Dense and 2D Convolutional layers, which are responsible for learning features and making predictions. In contrast, pooling and dropout layers focus on improving efficiency and reducing overfitting. Together, these components enable the Convolutional Neural Network (CNN) to effectively learn and generalize, making it highly capable of tasks like image recognition and classification.

**Improving Robustness and Generalization with Augmentation**

To make the model more robust and better at generalizing to new data, real-time data augmentation techniques were applied to the training set. These included rotations, shifts, flips, zooms, and shearing, which added variety to the dataset. By introducing this variability, the model was better equipped to handle unseen data. Training was carried out on the augmented dataset in batches of 64 and lasted up to 35 epochs. Validation data was used throughout to monitor the model’s performance and ensure it was improving as expected.

To avoid overfitting, early stopping was used to halt training when validation loss stopped improving. Additionally, model checkpointing was implemented to save and restore the weights from the best-performing epoch. Together, these methods ensured that the model learned effectively from the data while maintaining accuracy and reliability when making predictions on new images.

The training process required several adjustments and repetitions over the course of a week due to challenges, which will be discussed later in the presentation.

Finally, the Training-Validation Loss Trend highlights that after multiple refinements, the model was well-trained and demonstrated strong performance on unseen data.

**From Metrics to Visuals: Understanding Model Behavior**

* The model achieves 55% accuracy, meaning it correctly predicts the class for 55% of the test samples. While this is better than random guessing (equivalent to 10% accuracy for a dataset with 10 classes), it certainly has the potential to arrive at better results.

The classification metrics results, as seen on the right-hand side of this slide, indicate that the model performs unevenly across different classes. The model performs well on structured and easily distinguishable classes like **ship** (F1-score: 0.72) and **truck** (F1-score: 0.65), likely due to their clear patterns. In contrast, it struggles with less distinct or more complex categories like **cat** (F1-score: 0.32) and **bird** (F1-score: 0.35), where more subtle patterns make differentiation more challenging.

The macro average and weighted average results here, suggest that while the model is effective at recognizing certain well-defined objects, it has difficulty generalizing to more ambiguous categories. Possible ideas for improvement include further fine-tuning hyperparameters, or additional training with more diverse and representative samples. All this could help achieve better performance across all classes.

**Performance Overview: Consistent Accuracy with Notable Misclassifications in Similar Classes**

We have arrived at the third section, which is Model Optimization and Validation.

The model achieves a test accuracy of 55.22% and a loss of 1.24, aligning with validation trends and demonstrating consistent performance across datasets. While this indicates stable generalization, it is significantly worse than the state-of-the-art accuracy that falls anywhere between 90 and 99%. The confusion matrix reveals strengths in predicting classes like frog (6) and ship (8) but highlights struggles with visually or contextually similar categories, such as airplane vs. ship (0 vs. 8) and cat vs. frog (3 vs. 6), likely due to shared visual features.

This would be another area where the model could be tweaked to achieve better results. Some of these tweaks include a more diverse and larger datasets like CIFAR-100 to improve the model's ability to differentiate between classes. Hyperparameters such as batch size, learning rate, and number of epochs could be further explored and adjusted. Also, more advanced architectures could be used or just a higher number of filters in convolutional layers for more refined outcomes.

**Refining The Model: Tackling Overfitting with Data Split Adjustments**

The initial model faced challenges with overfitting and low validation accuracy due to an imbalanced 70/20/10 data split, too many epochs (50), and a batch size of 64. Early fixes included adding dropout and BatchNormalization layers, reducing the batch size to 32, and limiting epochs to 30. However, these adjustments caused unstable validation loss and worsened overfitting.

To fix these issues, the data split was changed to 80/10/10, creating a better balance between training, validation, and test sets. This adjustment laid the foundation for better performance. Next, the learning rate was reduced significantly (to 0.0003–0.0004), allowing the model to make smaller, more stable updates during training. The early stopping patience was increased to 5–7 epochs, giving the model extra time to improve without risking overfitting. Also, the Adam optimizer was selected because it updates weights efficiently, speeding up learning. Finally, random seeds were set to ensure consistent and reproducible results.

Additional improvements were made by setting the batch size back to 64, limiting training to 35 epochs, and applying data augmentation techniques such as rotations and flips. These techniques introduced variety to the training data, helping the model perform better on new, unseen data.

To better understand the dataset, a tool called utility function was used to display grids of images with their labels. The model’s architecture was also simplified by reducing the number of filters in the 2D convolutional layers and the number of neurons in the dense layers. These changes helped the model generalize more effectively and reduced the risk of overfitting.

By fixing the data split, fine-tuning hyperparameters, simplifying the architecture, and using visualization tools, the final model became more stable, efficient, and capable of performing well on unseen data.

Certain model design choices were intentionally simplified to prioritize achieving a stable, functional model over optimizing for peak performance. This approach was primarily influenced by limited programming expertise, as implementing and troubleshooting more complex adjustments proved time-intensive. Consequently, the convolutional layers were capped at three, the architectural complexity was kept minimal, and the addition of extra features was carefully chosen.

**Learnings and Conclusion**

Throughout the third module of this course, I saw significant growth in my Python programming skills, although I’m still at a beginner level compared to those who use it more frequently. Early on, I made an error with the test split, setting it at 70/20/10 instead of the correct 80/10/10. This mistake slowed progress and worsened results despite my adjustments. Once I identified and corrected the split, combined with reducing the epoch count, adding early stopping, and using the original batch size, the model became much more usable. Debugging was a key part of this process, involving careful checks of data splits, code validation, and using random seeds to ensure reproducibility.

Working in Python came with challenges, like slower debugging and fewer resources compared to platforms like Google Collab. But these obstacles ultimately deepened my understanding of neural networks—especially convolutional ones—and how architecture design can significantly impact outcomes. Through iterative refinements, including hyperparameter tuning, I not only improved the model but also gained valuable insights into the training and optimization process. Visualizing progress with graphs and metrics also helped reinforce my learning.

By addressing the test split issue and fine-tuning parameters, I was able to achieve stable and reliable results. This project not only improved my technical skills but also helped me better understand how to approach future improvements. It gave me a clearer idea of how Convolutional Neural Networks (CNNs) work in real-world applications, how they recognize images, and how predictions are made.

A potential continuation of this project could involve a few previously mentioned features. The model’s performance could be more powerful by using larger datasets like CIFAR-100, optimizing hyperparameters further such as batch size, learning rate, and epochs, and implementing advanced architectures or increasing convolutional filters for improved class differentiation and refined outcomes.