**Slide 1: Title Slide** Hello everyone and thank you for joining me today. My name is Majed Alzaabi, and this presentation is a culmination of my assignment titled *Neural Network Models for Object Recognition Using Multi-Track Machine Learning Approaches*. The focus of this project is the design, training, and evaluation of neural network models for object recognition using the CIFAR-10 dataset. We will explore how different ML tracks—Classical ML, Deep Learning, and Advanced ML—handle object recognition and discuss their strengths and limitations. This analysis helps us understand how various machine learning paradigms can be applied in real-world AI systems and prepare us for making informed model selections in practical applications.

**Slide 2: Project Overview** AI-driven object recognition has become an essential component in numerous applications across various industries. For instance, self-driving cars rely on object detection to identify pedestrians, vehicles, and road signs, which is critical for safety and navigation (Grigorescu et al., 2020). Face recognition systems, such as Apple’s Face ID, are widely used for authentication (Parkhi et al., 2015). Similarly, video surveillance systems use object recognition for activity monitoring and threat detection in crowded environments, airports, and smart cities.

In this project, we aim to construct and compare ML models trained on the CIFAR-10 dataset. CIFAR-10 is a popular benchmark dataset in computer vision consisting of 60,000 32×32 color images in 10 classes, with 6,000 images per class (Krizhevsky, 2009). These classes include everyday objects like cats, dogs, airplanes, ships, and trucks. It provides a manageable yet challenging benchmark for deep learning models.

We apply and compare models across three tracks:

* Track 1: Classical ML, involving feature engineering and traditional classifiers such as SVM and KNN.
* Track 2: Deep Learning, focusing on Convolutional Neural Networks (CNNs) and transfer learning.
* Track 3: Advanced ML, incorporating self-supervised learning and neural architecture search.

The goal is not only to evaluate performance but to gain insights into the trade-offs, limitations, and practical considerations of each approach.

**Slide 3: Data Preparation and Dataset Splitting** We begin by loading and preprocessing the CIFAR-10 dataset. The dataset is loaded using standard libraries such as TensorFlow or PyTorch, and the image pixel values are normalized to the [0, 1] range. This step improves the stability of gradient descent and accelerates convergence during training (Goodfellow et al., 2016).

The dataset was split as follows:

* 40,000 images for training
* 10,000 images for validation
* 10,000 images for testing

The training set is used to learn the model parameters, while the validation set is essential for tuning hyperparameters and monitoring model generalization during training. The test set remains untouched until the final evaluation to provide an unbiased measure of performance.

In addition, we reshaped the input images into a 4D tensor format and verified that the dataset contains a balanced distribution across all 10 classes.

**Slide 4: Validation Set Rationale** The validation set plays a crucial role in machine learning pipelines. It allows us to evaluate model performance on unseen data during training without contaminating the test set. For example, if a model achieves 95% accuracy on the training set but only 70% on the validation set, this signals overfitting, where the model memorizes training patterns but fails to generalize (Bishop, 2006).

To mitigate this, the validation set is used to guide adjustments in learning rate, network depth, regularization, and other hyperparameters. Furthermore, tools such as early stopping rely on validation performance to prevent overfitting by halting training once validation accuracy plateaus.

Using a dedicated validation set is standard practice in both academic and industrial machine learning workflows (Geron, 2019).

**Slide 5: Model Selection** For this project, we emphasized Track 2: Deep Learning using Convolutional Neural Networks. CNNs have revolutionized computer vision tasks by learning spatial hierarchies of features through convolution operations and local connectivity (LeCun et al., 2015). Unlike classical ML, which requires manual feature extraction, CNNs learn useful representations directly from the raw pixel data.

Track 1, Classical ML, employs models like Support Vector Machines (SVM) and K-Nearest Neighbors (KNN). While efficient for structured data, these models rely on hand-crafted features such as Histogram of Oriented Gradients (Dalal & Triggs, 2005) or color histograms. Their performance is limited on high-dimensional image data.

Track 3, Advanced ML, includes cutting-edge approaches such as SimCLR for self-supervised learning (Chen et al., 2020) and reinforcement-learning-based Neural Architecture Search (Zoph & Le, 2017). These methods have shown remarkable results in recent years but demand extensive computational power and expertise.

**Slide 6: Architecture & Hyperparameters** Our CNN model architecture is inspired by proven deep learning patterns and includes:

* Input: 32×32×3 image tensor
* Conv2D layer with 32 filters, kernel size 3x3, activation ReLU
* MaxPooling layer with 2x2 pool size
* Conv2D layer with 64 filters
* MaxPooling layer
* Conv2D layer with 64 filters
* Flatten layer to convert 3D data to 1D
* Dense layer with 64 units and ReLU
* Output Dense layer with 10 units and softmax activation

We used the Adam optimizer (Kingma & Ba, 2015), which combines the advantages of momentum and RMSProp. Hyperparameters included:

* Learning rate: 0.001
* Batch size: 64
* Epochs: 10

This architecture is compact yet powerful enough to achieve good accuracy on CIFAR-10 without overfitting. We intentionally avoided deeper networks like ResNet to limit training time and keep the implementation accessible.

**Slide 7: Performance Metrics** Performance evaluation was carried out using scikit-learn metrics (Pedregosa et al., 2011):

* **Accuracy**: Reached 70% on the test set
* **Precision**: High for structured classes like automobile and ship
* **Recall**: Balanced for most classes, but lower for ambiguous ones
* **F1 Score**: Good overall balance between precision and recall

We also generated a **confusion matrix**, which helped identify class-level errors. For instance, the model sometimes confused cats with dogs and birds with airplanes, likely due to similar visual textures in low-resolution images.

These results are consistent with findings in literature suggesting shallow CNNs struggle with fine-grained distinctions (He et al., 2016).

**Slide 8: Visual Results** To better understand training dynamics, we plotted accuracy and loss curves over epochs. These visualizations show that:

* Training accuracy increased steadily, reaching 74%
* Validation accuracy stabilized at 70%
* Training and validation losses decreased concurrently

No significant overfitting was observed, indicating that the model generalized well given its simplicity. The confusion matrix visualization confirmed that vehicles such as trucks and ships were easier to classify, while animals like cats and dogs posed more difficulty.

**Slide 9: Training Strategy** Our training strategy was intentionally simple to maintain clarity and reproducibility. It included:

* Adam optimizer for stable and adaptive learning
* Mini-batch training with batch size of 64
* 10 training epochs without early stopping

Despite being basic, this setup achieved a reasonable trade-off between performance and training time. We recommend future extensions with data augmentation, dropout layers, and learning rate schedulers to further improve performance.

For example, augmenting the dataset with rotated, flipped, and cropped images can improve model robustness. Similarly, using early stopping based on validation loss can prevent unnecessary training beyond optimal convergence.

**Slide 10: Comparative Discussion** Comparing all three ML tracks:

* **Track 1 (Classical ML)** is suitable for low-dimensional or structured data and offers interpretability. However, it is inadequate for raw image classification.
* **Track 2 (Deep Learning)** is a practical balance of accuracy and scalability. CNNs automatically learn spatial features, leading to better performance.
* **Track 3 (Advanced ML)** offers cutting-edge performance but is costly. Techniques like neural architecture search (NAS) and contrastive self-supervised learning (BYOL, MoCo) match or surpass supervised methods (Grill et al., 2020).

Ultimately, CNNs represent a powerful yet accessible solution for image classification and remain a cornerstone in modern AI systems.

**Slide 11: Conclusions** In conclusion, we:

* Designed and implemented a CNN for object recognition on CIFAR-10
* Achieved 70% test accuracy with a relatively simple architecture
* Explored and compared multiple machine learning approaches
* Evaluated our model using detailed performance metrics

This project enhanced our understanding of image classification, model tuning, and the strengths and weaknesses of different ML strategies. It also highlighted the importance of validation practices, performance analysis, and continual model improvement.

**Slide 12: Lessons Learned & Future Work** Key lessons include:

* CNNs are highly effective for learning from raw image data
* Validation sets and metric tracking are vital for tuning
* Classical ML is limited for high-dimensional visual tasks

For future work, we plan to:

* Implement dropout and batch normalization layers
* Explore transfer learning using MobileNet, ResNet, or EfficientNet
* Apply data augmentation to improve generalization
* Integrate NAS or AutoML tools to optimize architectures

These improvements could push accuracy beyond 90%, making the model more competitive in real-world tasks.

**Slide 13: References:**

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25, 1097–1105. https://doi.org/10.1145/3065386

11. Géron, A. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow (2nd ed.). O’Reilly Media.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Zoph, B., & Le, Q. V. (2017). Neural architecture search with reinforcement learning. International Conference on Learning Representations (ICLR). https://arxiv.org/abs/1611.01578

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. https://doi.org/10.1038/nature14539.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press. https://www.deeplearningbook.org/

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770–778. https://doi.org/10.1109/CVPR.2016.90

Chen, T., Kornblith, S., Norouzi, M., & Hinton, G. (2020). A simple framework for contrastive learning of visual representations. International Conference on Machine Learning (ICML), 1597–1607. https://arxiv.org/abs/2002.05709