Transcript - Object Recognition with CNNs (Sarah Thompson)

Hi, welcome to my presentation on object recognition using deep learning.
In this project, I compared two machine-learning approaches on the CIFAR-10 dataset. Firstly, a custom convolutional neural network trained from scratch and secondly a VGG16 transfer-learning model pretrained on the ImageNet dataset.

My goal was to understand how data preparation, network architecture, and pretrained knowledge can influence model accuracy.

Convolutional networks were first introduced by LeCun and his team back in the 1990s, forming the backbone of today's computer-vision systems. Work that followed, especially by Goodfellow and Bengio, showed how deeper layers and ideas like dropout could make these models much stronger and more reliable.

In the next few slides, I'll start by exploring the dataset and preprocessing steps, then walk through both models, their training results, and finally share the key insights and lessons I have learned from the experiment.

The CIFAR-10 dataset, first introduced by *Alex Krizhevsky and Geoff Hinton* in 2009, contains sixty thousand colour images, evenly spread across ten object categories, including airplanes, automobiles, cats, dogs, and several others.

Each image is just thirty-two by thirty-two pixels, which makes classification more challenging. There isn't much fine detail to rely on, so models need to learn robust, high-level features rather than memorising surface pattern.

CIFAR-10 is often described as a medium-complexity dataset, it's simple enough for experimentation and teaching, yet still complex enough to reveal real differences in how models learn and generalise. That balance is one reason it has become one of the most widely used benchmarks in computer-vision research.

For this project, I used forty-five thousand images for training, five thousand for validation, and ten thousand for testing. Keeping a separate validation set allows hyperparameter tuning and early stopping to reduce overfitting.

Before training, I normalised all images to a zero-to-one range and one-hot encoded the labels. Normalisation, ensures stable learning and faster convergence.

These preprocessing steps standardised the inputs and guaranteed that both models received identical and fair training data.

To improve model generalisation, I implemented a series of light data-augmentation transformations, horizontal flips, small rotations, translations, and zooms.
Each epoch, the model saw slightly different versions of each image, which increased variability and reduced overfitting.

Research shows that such realistic transformations enhance robustness by teaching the network to recognise objects from multiple viewpoints.

Both the baseline CNN and the VGG16 transfer-learning model used the same data preprocessing and augmentation setup so that any performance differences would reflect the architecture, not the data itself.

For the VGG16 model specifically, I first resized the images to bring them closer to the standard ImageNet input size. I stopped at ninety-six by ninety-six pixels, since increasing to the full two-twenty-four by two-twenty-four size caused Google Colab to crash due to memory limits.

This resizing step was done before scaling the pixel values and applying one-hot encoding, ensuring that all subsequent preprocessing operations remained consistent and stable across both models.

My first model was a convolutional neural network built entirely from scratch for the CIFAR-10 dataset.

It consisted of two convolutional blocks, meaning pairs of convolution and pooling layers, each using four-by-four filters to learn patterns like edges and textures, followed by max-pooling layers to reduce the image size while keeping the most important features.

After the layers, the output was flattened and passed through a dense layer with two-hundred-and-fifty-six neurons to capture higher-level representations, and then a softmax output layer produced the final ten-class predictions.

I trained the model using the Adam optimiser, with its default learning rate of one-times-ten-to-the-minus-three, and applied early stopping to prevent overfitting, halting training when the validation loss stopped improving.

To evaluate the impact of data augmentation, I trained two versions of this model: one using the original dataset, and one with an augmented dataset that included random flips, small rotations, and translations. This allowed me to clearly see how augmentation improved the model's ability to generalise.

This straightforward design follows the classic architecture principles for small-image classification tasks. It's compact, easy to interpret, quick to train, and provides a strong baseline for comparison with the VGG16 transfer-learning model.

This slide shows the model summary for my custom convolutional neural network.

It walks through each stage of the architecture, the convolutional blocks that extract features, the pooling layers that reduce spatial dimensions, and finally the flatten and dense layers that perform classification.

At the bottom, you can see the total number of trainable parameters, which highlights how compact and efficient this model is compared to larger architectures.

The second model applied transfer learning using the VGG16 architecture, originally introduced by Simonyan and Zisserman in 2015.

Instead of learning from scratch, this model reused convolutional filters pretrained on the large-scale ImageNet dataset, which contains over fourteen million images.
These pretrained filters already capture fundamental visual patterns, such as edges, colours, and textures.

I removed the original ImageNet classifier and added a new custom head explicitly designed for CIFAR-10.

It followed the same overall structure as my earlier CNN, using global average pooling to reduce spatial dimensions, a dense ReLU (ray-loo) layer with 256 units, dropout set to 0.5 for regularisation, and a softmax output for the ten classes.

This was intentional, because I wanted to mirror my baseline CNN's classifier design so that differences in performance could be attributed to the pretrained convolutional base rather than differences in architecture.

During training, I kept the early convolutional layers frozen so they'd retain their general feature knowledge from ImageNet, and only fine-tuned the final layers to help the model recognise the unique patterns in CIFAR-10.

This table shows the model summary for the VGG16-based transfer-learning model. It lists the layers inherited from the original ImageNet-trained VGG16 architecture, followed by the custom classifier I added for CIFAR-10.

You can see that most of the parameters come from the frozen convolutional base, while only a smaller portion, the new top layers, are trainable.

This structure follows the same design principles as the original architecture, allowing the model to reuse powerful pretrained features while adapting effectively to a new, smaller dataset.

To stabilise training, I used early stopping, which halts the process when validation performance stops improving.

Initially, the patience parameter was set to two epochs, but I found that for models two and three, training sometimes stopped too early, before reaching the lowest loss and highest accuracy.
So, I fine-tuned the patience setting to five, allowing the model a few extra epochs to recover and continue improving.

I also used TensorBoard throughout training to visualise accuracy and loss in real time.
This helped me compare the learning curves for all three models.

Here we can see the training and validation accuracy for the three models.

The CNN without augmentation improved quickly but began to overfit after around ten epochs, shown by the widening gap between the curves.

The augmented CNN improved more steadily, maintaining alignment between training and validation accuracy for longer.

VGG16 achieved the fastest convergence and the highest accuracy overall.

Looking at the loss curves, the same pattern appears.

The baseline CNN's validation loss started increasing early, signalling overfitting.

The VGG16 model reached a lower and more stable validation loss overall, illustrating its stronger generalisation.

As noted by Goodfellow and Bengio, monitoring these metrics is essential for diagnosing when a model shifts from learning to memorising.

After training, each model was evaluated on the test set of ten thousand images using accuracy, precision, recall, and F1-score.

The CNN without augmentation had the weakest performance, while the augmented CNN achieved moderate improvements.

The VGG16 transfer-learning model achieved the best overall metrics, with accuracy roughly five to ten percent higher.

Starting with the baseline CNN, we can see that it struggled with visually similar classes, particularly between cats and dogs, and also across some of the vehicle categories.

Its predictions are more widely spread away from the diagonal, which represents correct classifications.

When I introduced data augmentation, the balance across classes improved noticeably. You can see more predictions aligning along the diagonal and fewer large off-diagonal errors.

This demonstrates how augmentation enhances robustness and helps models generalise better, even when training data is limited.

Finally, the VGG16 transfer-learning model shows the cleanest diagonal pattern of all three, with the fewest misclassifications. This reflects its stronger ability to distinguish between similar classes, confirming the benefit of pretrained feature representations.

Together, these three matrices clearly show the progression in performance, from a simple model that overfits, to one improved by augmentation, to a pretrained model capable of high-level feature separation.

Here we can directly compare the quantitative results for all three models.

Starting with the CNN trained without data augmentation, overall performance was the lowest, with precision, recall, and F1-scores averaging around 0.67. This pattern suggests clear overfitting, where the network memorised the training data but struggled to generalise to unseen examples.

When data augmentation was added, the model's weighted averages increased to 0.71, showing a clear improvement across most classes. Precision and recall became more balanced, which means the model was better at both identifying true positives and avoiding false ones.

Finally, the VGG16 transfer-learning model achieved the strongest overall results, with precision, recall, and F1-scores all close to 0.78–0.79. This indicates a consistent boost in predictive performance across all categories.

In particular, VGG16's higher recall reflects its ability to correctly detect objects more reliably, a direct result of leveraging pretrained features learned from the large-scale ImageNet dataset.

Across all three, the progression from **0.67 to 0.71 to 0.78** demonstrates the combined benefits of data augmentation and transfer learning.

Comparing both models reveals a clear balance between simplicity and performance.

The custom CNN is light, quick, and flexible; however, its limited depth restricts feature extraction.

The VGG16 model achieved higher accuracy and faster convergence, but at the cost of computational intensity and reduced architectural flexibility.

This trade-off mirrors the efficiency-versus-adaptability discussion by Howard and Ruder.

This project really deepened my understanding of how convolutional neural networks learn and generalise.

I saw first-hand how a proper validation set is crucial for detecting and controlling overfitting, while techniques like dropout and early stopping help stabilise learning and improve generalisation.

Data augmentation stood out as one of the most effective strategies. By slightly transforming the training images, I increased data diversity and reduced overfitting.

I also learned that more training isn't always better. Early overfitting in my CNN showed the importance of regularisation and knowing when to stop rather than chasing higher training accuracy.

Comparing from-scratch learning with transfer learning reinforced how pretrained models like VGG16 can learn far more efficiently and achieve higher accuracy.

This project took me beyond simply building neural networks to really understanding why they behave the way they do, and how architectural choices, data, and optimisation are all vitally important.

Convolutional neural networks now underpin a wide range of real-world AI systems, from autonomous vehicles and facial recognition to diagnostic tools in healthcare. Studies show that CNNs can achieve dermatologist-level accuracy in skin-cancer detection, highlighting the immense potential of image-based deep learning when applied responsibly.

The techniques explored in this project, data augmentation, validation-driven training, and transfer learning, all contribute to building models that are both accurate and generalisable in practice.

But with this potential comes responsibility. Responsible AI must prioritise fairness, transparency, and inclusiveness.
It's not just about performance metrics; it's about ensuring that models behave ethically across diverse contexts.

There is also a vert real concern about the environmental cost of large-scale models, computational efficiency is not only a technical goal, but an ethical necessity.

Looking ahead, there are several directions I'd like to explore.

For the VGG16 model, I plan to experiment with different normalisation techniques, such as applying dataset-specific mean and standard deviation scaling as outlined in several other implementations, and resizing the images to the full 224×224 input expected by ImageNet models for improved alignment.

I am also interested in experimenting with newer architectures like ResNet50 and MobileNetV2.

To conclude, both models successfully recognised objects in the CIFAR-10 dataset. The custom CNN provided a strong and interpretable foundation, while the VGG16 transfer-learning model achieved higher accuracy and faster convergence by building on pretrained knowledge.

Together, they capture the shift from feature learning to transfer learning, reflecting the ongoing evolution of deep learning itself.

Thank you very much for listening.