Audio transcript – development individual project

Slide 1

Hi. For this final assignment of the machine learning module, I will present a development project on neural networks for object recognition.

Slide 2

This is the agenda for this presentation.

Slide 3

Artificial intelligence is increasingly being used to extract information from varied sources, including images.

Computer vision models can have multiple applications, such as face recognition, autonomous vehicle driving, and medical image interpretation (Chang *et al.*, 2018; Sarki *et al.*, 2022; Augmented AI, 2023; ubiAI, 2023).

These models can have multiple architectures and complex design features, and are challenging to develop and train. It is therefore increasingly important to understand how to compile, train, and tune these models.

Slide 4

This assignment required designing a convolutional neural network model for image recognition using the CIFAR-10 dataset (Krizhevsky, 2009a, 2009b). This dataset includes 60,000 web-scraped images, manually annotated across 10 mutually exclusive categories. The data were first pre-processed by normalising pixel values.

Slide 5

The next step was dataset partitioning. This is required so that the model does not provide overly optimist performance estimates by being trained on the same data used to test performance, and is instead tested on unseen data, which better mimics real world use (Bronshtein, 2020).

In the Python Keras module, the dataset is already partitioned into train and test sets in 5:1 to ratio, which I've kept (*Keras: Deep Learning for humans*, no date).

The training set also needs to be split into training and validation sets. As shown on the right, lowering prediction error on the train set can lead to model overfitting as the model begins to replicate random noise in the data, rather than truly informative features. This is then reflected on increasing prediction error on the validate set, in what is known as the bias-variance trade-off (Natras, Soja and Schmidt, 2022).

The training set is used to train the mode and the validation set to assess the performance of the candidate model in each training epoch. Again, the aim is to see how the model performs in data it has not yet seen as we go, so that we can take forward the candidate model with the best accuracy and test its performance on the test set.

The train-validate split can be performed in many ways (Berrar, 2019). In cases with lower data availability, cross-validation approaches such as k-fold cross validation can provide stable performance estimates while ensuring the model is trained on as much data as possible. However, if data size is large, such as tens of thousands in our case, the simple hold-out method can be used in the same way as the train-test split, and is more computationally efficient. There are no clear rules for splitting the train and validation set, but given the large amount of data a standard 80/20 split should be appropriate and is a commonly used approach. This leads to the 40,000 – 10,000 split applied.

You can see below some snippets of my code. I'll include these along my presentation for reference, but won't be describing them.

Slide 6

For this image classification task, I will be using a convolutional neural network (Krizhevsky, Sutskever and Hinton, 2012; Swapna., 2020). Generally speaking, the first section is composed of convolutional layers, where a set of square kernels or filters are moved along a numerical representation of the input figure.

The numerical inputs are processed using the weights contained in each kernel in consecutive layers, allowing the model to learn important features in the input without being explicitly told which.

Included in this architecture are also activation layers, which determine whether information from a particular node or kernel is passed to the next layer or discarded.

Outputs from several nodes are then aggregated via pooling layers, which progressively reduce output dimensions and allow the model to learn more complex features.

The final outputs from the convolutional layers are then flattened into a single vector and passed through a set of fully connected layers within a conventional artificial neural network, which learns how to classify each input into a set of possible target classes.

Slide 7

I tested two main network architectures to assess the impact of model complexity on training requirements and performance.

The first is what I called a simple CNN. This included only 2 convolutional layers and 2 dense layers, with a total of 225 thousand parameters.

The second was the VGG-19 network, an architecture first proposed in 2014 and commonly used for image classification tasks (Simonyan and Zisserman, 2015). This

network includes a total of 19 convolutional and dense layers and close to 40 million parameters.

Slide 8

Here you can see the technical implementation of each model in Python. Let's now delve a bit further into some architecture choices.

Slide 9

Activation functions within CNNs allow the model to learn complex features by introducing non-linearity and neuron firing thresholds (Bag, 2023). The RELU function is commonly used in CNNs as it discards information for outputs lower than 0, creating sparsity and facilitating learning. It also helps avoid vanishing gradients when compared with a standard sigmoid function, since learning does not plateau with increasingly positive inputs (Srivastava, 2024).

As for the final activation function, the softmax function is a common choice for CNNs as it allows the model to recognise and leverage the inherent uncertainty in the predictions it makes (Belagatti, 2024). Therefore it can learn not only from the final prediction it makes, but also from the degree of certainty or probability it allocates to each possible option.

Slide 10

For the loss function, I've chosen the sparse categorical cross-entropy function. This is a standard choice for multi-class classification problems and is more efficient than binary categorical cross-entropy loss, as each class is coded with an integer rather than one-hot encoding (Fortuner, 2017; Hughes, 2024).

This function is particularly advantageous as loss increases exponentially when low probabilities are attributed to the correct class, and decreases exponentially if higher probabilities are assigned to the correct class.

Slide 11

The optimizer is the algorithm which specifies how which set of weights the model will choose to test after each training batch. The optimizer tries to find the global loss function minimum as quickly as possible without becoming trapped in local minima.

Several algorithms have been proposed, of which stochastic gradient descent (or SGD) and the Adaptive Moment Estimation (or Adam) are common examples (Jiang, 2020; Musstafa, 2022).

SGD is a simple and non-adaptive method, but which has been suggested to have better generalisability than more complex methods, including in the CIFAR-10 dataset (Lu, 2017).

By contrast, Adam is a complex adaptive approach that combines features from Adaptive Gradient and Root Mean Square Propagation algorithms - that is, it is able to retain information about the degree and direction of previous weight updates and use this as momentum, and also the direction of descent along the weight domain so that domain areas not yet explored can be tested as well (Kingma and Ba, 2017).

Adam has a generally superior performance on large datasets, and is more computationally efficient, but may generalise worse than SGD (Wilson *et al.*, 2018; Giordano, 2020; Adhikari, 2023). Therefore I tried using both algorithms in the two network architectures described earlier.

Slide 12

Finally, this is the general training approach I employed:

- Batch size was set at 64 (that is, the number of instances ran through the network to calculate the loss function each time the weights are updated). Lower batch sizes generally increase training time per epoch, but lower the total number of epochs and training time required for convergence. 64 is a standard batch size for CNNs that generally balances training time and stability (Chang and Pathak, 2020)
- The maximum number of training epochs was set at 50. Preliminary testing suggested that a much lower number of epochs would be generally sufficient for model convergence, but I wanted to allow more complex models to have a reasonable chance of improving prediction accuracy.
- The callback or early stopping rule was initially set as two consecutive increases in validation loss, as I thought it would be unlikely for this to represent a local minimum and wanted to keep training relatively efficient.
- And the learning rate was set as the default for each optimizer algorithm

Slide 13

Let's now start delving into comparative model performance. We can see here that for the simple model, the Adam algorithm yielded higher training accuracy and more efficiently than SGD, but resulted in similar validation accuracy.

When we switch to VGG-19, we see that both training and validation accuracy increase significantly, with training time and number of epochs decreasing versus simple CNN with SGD.

But then, when we try to train VGG-19 with Adam, we see a very strange behaviour – the maximum achieved accuracy is 9% in both train and validation sets, with training stopping after only 3 epochs due to the early stopping rule. It seemed that the model was not being able to learn, and this behaviour was consistent despite dozens of attempts. I did not understand what this meant, despite spending days attempting multiple troubleshooting approaches.

Slide 14

I eventually decided to resort to ChatGPT for help.

It agreed with my intuition that this was an issue with learning, and

provided what seemed to be some reasonable suggestions for the underlying root causes and possible solutions. I could have taken the code it suggested and applied all together, but I wanted to explore the rationale and impact of each of these.

Slide 15

The first I tried was reducing the learning rate to stabilise training. This immediately allowed the model to achieve very reasonable train and validation accuracies, but it seemed that it was possibly learning too quickly as the early stopping rule was being met very fast. However further reducing the learning rate did not make a difference here.

At the same time, from the shape of the accuracy and loss plots, it seemed that the model was already starting at a point where complexity was already too high and leading to overfitting (Zulkifli, 2018; Kurban, 2021).

Slide 16

The next suggested option was to use batch normalisation to ensure node outputs were appropriately scaled across filters within the same batch. This seemed to slow down and stabilise learning, but initial model complexity still seemed too high.

Slide 17

Next, I tried using dropout layers, where a proportion of all neurons within the fully-connected layers are shut down to discard some information and avoid overfitting. This seemed to also allow more stable training, but still did not resolve the high initial complexity issue.

Slide 18

The final option was to greatly reduce the number of nodes in the fully-connected layers. This allowed the model to converge significantly faster than previous iterations, while achieving similar performance.

Slide 19

For completion, I also tried adding together dropout layers, but this did not have any impact on training or performance.

I had now settled on a model and training approach I was generally satisfied with, but I still wanted to explore a couple of other approaches I had come across to learn more about them and try to improve model performance.

Slide 20

The first was employing a transfer learning technique by using pre-trained weights for the CNN section of VGG-19, based on the ImageNet dataset (Solawetz, 2021; Stanford Vision Lab, Stanford University, Princeton University, 2021).

I was surprised to see that this not only did not improve, but it significantly worsened performance and increased training time versus training the model from scratch, perhaps as the pre-learned weights were developed in a dataset that was much more complex and too different from the one being used here.

Slide 21

Finally, I tried using data augmentation, where the input data are subject to a series of transformations to add noise and force the model to learn more representative features. This should in principle lead to worse training performance, but improved validation performance.

What I saw was that although loss functions seemed to be going in the right direction and speed, the model stopped training too early.

Not sure what this meant, I decided again to ask ChatGPT for some help.

Slide 22

It thought the issue was a mix of applying too complex augmentation transformations for the data at hand, and becoming trapped in local minima due to the learning rate being too low.

Slide 23

I accepted his suggestions and tried smoothing my data augmentation approach. This helped somewhat, but learning once again stopped too early.

Slide 24

The next suggestion was implementation of a learning rate scheduler. This combines an adaptive learning rate with a more permissive callback, so that the learning rate was halved if validation loss increased for 3 consecutive epochs.

We can see here that learning was immensely stabilised by this approach, starting at much higher training loss, slowly progressing through the weight domain, and effectively stopping after about 15 epochs.

Slide 25

This table shows the final performance metrics for all models trained.

We can see that the simple CNN models had much lower performance than VGG-19,

but similar to VGG-19 if trained using pre-trained weights and harsh data augmentation.

Most candidate VGG-19 models had modest performance,

except those where model complexity was curtailed,

and then further improved with smooth data augmentation and an adaptive learning rate.

Given these results, model 13 was selected as the final model for assessment in the test set.

Slide 26

Overall precision, recall, F1-score, and accuracy were 0.88. This performance is far from perfect, but mostly similar to results published in the computer science literature (Gouk *et al.*, 2018; Andrade, 2019). Classes with best performance were those related to inanimate objects, possibly as these have more distinctive shapes, with the most common misclassifications occurring between cat and dog or frog (Krizhevsky, 2009a).

Slide 27

This brings me to my learning points and conclusions from this assignment.

First, I saw first-hand the importance of network complexity and depth on prediction accuracy,

but conversely the negative impact of overfitting and the need for regularisation techniques to ensure that model complexity remains appropriate to the specific task

I explored the impact of different optimizer algorithms on learning stability and speed, and also the importance of carefully selecting an appropriate learning rate for each specific model.

I also learned that transfer learning may not always be a useful approach,

and similarly that data augmentation can be helpful but must be done carefully.

and finally, I learned the hard way about balancing model complexity and tuning with computational requirements, and associated costs

Slide 28

In conclusion, in this assignment I tested the impact of different CNN architectures and training approaches for a simple image classification task,

and in doing so, learnt about many common machine learning challenges and how to overcome them.

The resulting model had a reasonable performance on the CIFAR-10 dataset,

which could have been further improved with finer hyperparameter tuning, including different kernel dimensions, stride, and padding, but the available computing capacity was limited for this.

More broadly, I got to experience what I felt was a genuine AI feedback loop, in which ChatGPT is used to troubleshoot development of a CNN, which in turn can be used to parse visual inputs to train ChatGPT and improve its capabilities, in either a virtuous or vicious cycle.

Slide 29

These are my references.

Slide 30

Thank you for your attention.

References

Adhikari, T. (2023) 'Designing a Convolutional Neural Network for Image Recognition: A Comparative Study of Different Architectures and Training Techniques'. Rochester, NY: Social Science Research Network. Available at: https://doi.org/10.2139/ssrn.4366645.

Andrade, A. de (2019) 'Best Practices for Convolutional Neural Networks Applied to Object Recognition in Images'. arXiv. Available at: https://doi.org/10.48550/arXiv.1910.13029.

Augmented AI (2023) *How to Implement Object Detection Using Deep Learning: A Step-by-Step Guide*. Available at: https://www.augmentedstartups.com/blog/how-to-implement-object-detection-using-deep-learning-a-step-by-step-guide.

Bag, S. (2023) 'Activation Functions — All You Need To Know!', *Analytics Vidhya*, 2 January. Available at: https://medium.com/analytics-vidhya/activation-functions-all-you-need-to-know-355a850d025e (Accessed: 20 January 2025).

Belagatti, P. (2024) *Understanding the Softmax Activation Function: A Comprehensive Guide*, *SingleStore*. Available at: https://www.singlestore.com/blog/a-guide-to-softmax-activation-function/ (Accessed: 20 January 2025).

Berrar, D. (2019) ‘Cross-Validation’, in *Encyclopedia of Bioinformatics and Computational Biology*, pp. 542–545. Available at: https://www.sciencedirect.com/topics/medicine-and-dentistry/cross-validation (Accessed: 20 January 2025).

Bronshtein, A. (2020) *Train/Test Split and Cross Validation in Python*, *Medium*. Available at: https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6 (Accessed: 17 January 2025).

Chang, D. and Pathak, A. (2020) *Effect of Batch Size on Neural Net Training | by Daryl Chang | Deep Learning Experiments | Medium*. Available at: https://medium.com/deep-learning-experiments/effect-of-batch-size-on-neural-net-training-c5ae8516e57 (Accessed: 20 January 2025).

Chang, P.D. *et al.* (2018) ‘Hybrid 3D/2D Convolutional Neural Network for Hemorrhage Evaluation on Head CT’, *American Journal of Neuroradiology*, 39(9), pp. 1609–1616. Available at: https://doi.org/10.3174/ajnr.A5742.

Fortuner, B. (2017) *Loss Functions — ML Glossary documentation*. Available at: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html (Accessed: 20 January 2025).

Giordano, D. (2020) *7 tips to choose the best optimizer*, *Medium*. Available at: https://towardsdatascience.com/7-tips-to-choose-the-best-optimizer-47bb9c1219e (Accessed: 20 January 2025).

Gouk, H. *et al.* (2018) ‘MaxGain: Regularisation of Neural Networks by Constraining Activation Magnitudes’. arXiv. Available at: https://doi.org/10.48550/arXiv.1804.05965.

Hughes, C. (2024) ‘A Brief Overview of Cross Entropy Loss’, *Medium*, 25 September. Available at: https://medium.com/@chris.p.hughes10/a-brief-overview-of-cross-entropy-loss-523aa56b75d5 (Accessed: 20 January 2025).

Jiang, L. (2020) *A Visual Explanation of Gradient Descent Methods (Momentum, AdaGrad, RMSProp, Adam)*, *Medium*. Available at: https://towardsdatascience.com/a-visual-explanation-of-gradient-descent-methods-momentum-adagrad-rmsprop-adam-f898b102325c (Accessed: 20 January 2025).

Keras: Deep Learning for humans (no date). Available at: https://keras.io/ (Accessed: 17 January 2025).

Kingma, D.P. and Ba, J. (2017) ‘Adam: A Method for Stochastic Optimization’. arXiv. Available at: https://doi.org/10.48550/arXiv.1412.6980.

Krizhevsky, A. (2009a) *CIFAR-10 and CIFAR-100 datasets*. Available at: https://www.cs.toronto.edu/~kriz/cifar.html (Accessed: 17 January 2025).

Krizhevsky, A. (2009b) ‘Learning Multiple Layers of Features from Tiny Images’, in. Available at: https://api.semanticscholar.org/CorpusID:18268744.

Krizhevsky, A., Sutskever, I. and Hinton, G.E. (2012) ‘ImageNet Classification with Deep Convolutional Neural Networks’, in *Advances in Neural Information Processing Systems*.

Curran Associates, Inc. Available at: https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (Accessed: 20 January 2025).

Kurban, R. (2021) *Boost your Network Performance*, *Medium*. Available at: https://towardsdatascience.com/boost-your-network-performance-cc0a2a95c5ef (Accessed: 20 January 2025).

Lu, S.-A. (2017) 'SGD > Adam?? Which One Is The Best Optimizer: Dogs-VS-Cats Toy Experiment', *SALu*, 28 May. Available at: https://shaoanlu.wordpress.com/2017/05/29/sgd-all-which-one-is-the-best-optimizer-dogs-vs-cats-toy-experiment/ (Accessed: 20 January 2025).

Musstafa (2022) 'Optimizers in Deep Learning', *Medium*, 12 February. Available at: https://musstafa0804.medium.com/optimizers-in-deep-learning-7bf81fed78a0 (Accessed: 20 January 2025).

Natras, R., Soja, B. and Schmidt, M. (2022) 'Ensemble Machine Learning of Random Forest, AdaBoost and XGBoost for Vertical Total Electron Content Forecasting', *Remote Sensing*, 14(15), p. 3547. Available at: https://doi.org/10.3390/rs14153547.

Sarki, R. *et al.* (2022) 'Automated detection of COVID-19 through convolutional neural network using chest x-ray images', *PLOS ONE*, 17(1), p. e0262052. Available at: https://doi.org/10.1371/journal.pone.0262052.

Simonyan, K. and Zisserman, A. (2015) 'Very Deep Convolutional Networks for Large-Scale Image Recognition'. arXiv. Available at: https://doi.org/10.48550/arXiv.1409.1556.

Solawetz, J. (2021) *An Introduction to ImageNet*, *Roboflow Blog*. Available at: https://blog.roboflow.com/introduction-to-imagenet/ (Accessed: 20 January 2025).

Srivastava, S. (2024) 'Understanding the Difference Between ReLU and Sigmoid Activation Functions in Deep Learning', *Medium*, 18 April. Available at: https://medium.com/@srivastavashivansh8922/understanding-the-difference-between-relu-and-sigmoid-activation-functions-in-deep-learning-33b280fc2071 (Accessed: 20 January 2025).

Stanford Vision Lab, Stanford University, Princeton University (2021) *ImageNet*. Available at: https://image-net.org/index.php (Accessed: 20 January 2025).

Swapna. (2020) *Convolutional Neural Network | Deep Learning*, *Developers Breach*. Available at: https://developersbreach.com/convolution-neural-network-deep-learning/ (Accessed: 20 January 2025).

ubiAI (2023) *Introduction to the COCO Dataset*. Available at: https://ubiai.tools/introduction-to-the-coco-dataset/.

Wilson, A.C. *et al.* (2018) 'The Marginal Value of Adaptive Gradient Methods in Machine Learning'. arXiv. Available at: https://doi.org/10.48550/arXiv.1705.08292.

Zulkifli, H. (2018) *Understanding Learning Rates and How It Improves Performance in Deep Learning*, *Medium*. Available at: https://towardsdatascience.com/understanding-

learning-rates-and-how-it-improves-performance-in-deep-learning-d0d4059c1c10 (Accessed: 20 January 2025).

References: