SUMMATIVE ASSESSMENT: INDIVIDUAL PRESENTATION

Assignment Topic: Neural Network Models for Object Recognition Using Multi-Track ML Approaches

1. Introduction

Computer vision is a field of Artificial Intelligence (AI) created with the purpose of training the computer to interpret and identify image properties, named features, followed by eventually reconstructing them (Fernandes, Dórea and Rosa, 2020). An important part of this field is represented by image classification. The general idea of this method is to input a set of pictures and train the computer to predict which class it belongs to, and it does this by analyzing and identifying features like shapes, textures or edges.

The subject of the following project is classifying images from the CIFAR-10 dataset using CNNs alongside a pretrained dataset, specifically MobileNetV2. The goal is to evaluate model performance using a series of metrics such as accuracy, precision and confusion matrices, and also experimenting with training configurations like epochs, learning rate and batch sizes.

2. Convolutional Neural Networks

A key player in the advancements of image classifications is deep learning, with Convolutional Neural Networks (CNNs) becoming a powerful tool for solving such tasks. This algorithm works by sliding, or convolving, a filter through an input image and multiplying its filter with the pixel values of the image. CNNs use multiple random filters, each eventually specializing to identify a certain type of pattern. The result is a feature map, which is then used for classification.

CNNs can analyze a variety of data types but have been especially good at image classification. By analyzing the current literature, we can see that this algorithm is superior to classical ML techniques or ANNs, showing a greater accuracy overall (Hasan et al., 2019; Salim and Mohammed, 2024) if they are provided with enough data.

3. CIFAR-10 Dataset

The CIFAR-10 dataset is widely popular, frequently used in machine learning and computer vision, alongside other datasets like ImageNet, which is the basis of the chosen pretrained model that I used. Collected by Alex Krizhevsky and Geoffrey Hinton (2009) it consists of 60.000 color images belonging to 10 labels or classes, each class containing 6.000 images of 32x32 pixels in size. The dataset is split into 50.000 images for training and 10.000 images for testing. It contains a balanced number of 10 labels, called classes, representing various shapes. The labels include objects like airplane, cat, deer, dog and truck.

4. ML technique

In modern deep learning, a pre-trained model like VGG16 or MobileNetV2, which has already learned how to extract meaningful patterns, is stripped of its head and the new model created is trained on top of the pre-trained base to match our specific dataset.

Initially, the pre-trained model chosen was VGG16 (Simonyan and Zisserman, 2015), a larger model with over 130 million parameters and many layers, that uses a simple architecture consisting of repeated 3x3 convolutions followed by max-pooling layers. This type of model was chosen thinking that by freezing most of it and fine-tuning, the result would be more accurate, but it ended up being very resource consuming, slow and with a tendency for overfitting.

Other options were taken into consideration and ultimately the model selected was MobileNet V2, a modern model shown to have great performance and offers quicker results with less computational efforts (Kumar Shukla and Kumar Tiwari, 2023).

5. Data Preprocessing

This task makes use of the popular deep learning library TensorFlow. To begin, the CIFAR-10 dataset was loaded using Keras. Loading the labels from Keras means that the order of classes is known, the exact ones being airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck. Flattening the labels is crucial because, when loading the dataset, each label is an array in itself. By flattening, 1D integer arrays were created, which is needed for proper TensorFlow and NumPy function performance.

The training set was split further into training and validation sets using a classic 80%-20% split, an important step for monitoring performance during training, especially since with smaller datasets like CIFAR-10 there is a tendency to overestimate accuracy (Vabalas et al., 2019).

Normalization was applied to the image pixel values. As a result, all pixel intensities originally ranging from 0 to 255 were scaled down to a range between 0 and 1 for better convergence during training. Additionally, the pre-trained models make use of normalized data, so unnormalized images are incompatible with their learned filters. The images were then resized from their original 32x32 resolution to a 96x96 resolution, to match the input requirements of MobileNetV2. A smaller resolution was chosen to limit memory use and speed up training without compromising model performance.

For the loss function the categorical crossentropy was chosen, since it is widely used and shows good performance. Because of this, one-hot encoding was applied to the labels, as the model calculates probabilities across multiple classes, and the categorical crossentropy loss function uses one-hot encoded vectors.

6. Model Architecture

The MobileNetV2 base was loaded excluding its final classification layers. Since this model is already trained on a large dataset (ImageNet), those learned features need to be kept intact, hence freezing the final layers.

To this base, a pooling layer was added. This particular layer compresses each one of the feature maps to a single number, an alternative to picking the widely used maximum or average value from small patches.

Furthermore, a dropout layer was added to prevent overfitting, alongside a dense layer with 256 units or neurons and a ReLU activation function. For the final dense layer, softmax activation was used, since this is a multi-classification case.

7. Training Configuration

To explore the impact of training parameters, the following configurable variables were defined. Batch size represents the number of processed samples before updating filter weights, and the

values 32 and 64 were used for fine-tuning. Epochs represent the number of passes through the entire training dataset and multiple values ranging from 10 to 25 were used.

Another parameter was the number of nodes in the custom dense layer, varying between 128 and 256. And finally, one of the most important parameters that defines the step size at each iteration while moving towards a minimum of the loss function, the learning rate. It was adjusted during the multiple executions set for fine-tuning, starting with the default of 0.001 coming from the Adam optimizer and experimented with 0.0001 and 0.0005.

Using the Adam optimizer with a custom learning rate allowed convergence fine-tuning, meaning that the efficiency and accuracy of learning was slightly improved over the default value. As previously mentioned, the model was compiled with categorical crossentropy as the loss function, and finally the choice for the main metric was accuracy.

8. Model Training and Evaluation

Training was performed on the prepared training and validation datasets using the Keras API. Training and validation metrics were continuously monitored to ensure the model did not overfit the data.

After training, the model was evaluated on the test dataset. This provides an idea of the performance of the freshly trained model. For more depth of analysis, some predictions on class probabilities were made and then the most likely class labels were derived.

Various evaluation metrics for measuring the model performance were used, among these being accuracy, meaning the percentage of correctly predicted labels, and also the confusion matrix, comparing training versus validation loss and accuracy. Based on the confusion matrix, for clarity and to add some aiding visual elements, a basic heatmap was created using Seaborn and Matplotlib.

9. Results and Visualisation

The best-performing model used the MobileNetV2 architecture with the following configuration: learning rate of 0.0001, batch size of 64, 25 epochs, and 256 dense units. This setup produced a test accuracy of 82.56%, with the initial step starting at 43.3% and reaching a score of 87.55% on the last epoch. While it is far from perfect, this result is solid considering the training constraints.

The plotted loss and accuracy curves were used to visualise learning progression. In the beginning, using the VGG16 pre-trained base, it was apparent from the first trial that the improvement after each step was very small, and the final accuracy was only 69%. This could be seen in both accuracy and loss curves, both hitting plateau quite early on. With the MobileNetV2, the plots showed that the model improved smoothly with minimal signs of overfitting. The results were slightly better when trained with a smaller learning rate, which is why the final value ended up being 0.0001.

The heatmap of the confusion matrix was created using a magma colour palette, more of a personal preference, but visually striking and a good choice for distinction. Looking at the

predicted versus true labels, some classes outperformed others. The lowest value corresponds to class 5, represented by the label “dog”, and it was frequently misclassified as class 3, represented by the label “cat”. Similarly, although at a smaller rate, class 4 which is the label “deer” was misclassified as class 2, which is the label “bird”. It is interesting to observe that misclassifications occur mostly with the same “type” of objects, meaning that the model possibly learned to discern between metal and organic matter through texture.

10. Challenges and Potential Improvements

From the beginning the biggest challenge was using VGG16, which caused out-of-memory (OOM) errors due to its high computational demands coupled with the large 224x224 image size requirement. This was supposed to be resolved by using Google Colab, since my own computer’s specifications proved to be poor, but even by using a more powerful GPU the issue was not solved. The final solution was to downsize the input images to 96x96 and switch to MobileNetV2.

This change was beneficial, but although switching to MobileNetV2 was more efficient, the multiple small fixes used for fine-tuning lowered the overall chances of achieving higher accuracy.

Possible improvements might include selectively unfreezing the last few layers of the pre-training model, making more room for MobileNet to adapt to the CIFAR-10 dataset. Another option that would possibly work is data augmentation. This technique involves applying transformations like random cutouts, rotation, and noise to the images in the dataset. Research has shown that data augmentation could make the model learn more efficiently, because increasing the diversity of the training data improves model generalization (Kumar et al., 2024).

A final example would be using techniques such as exponential decay or learning rate reduction instead of the static approach. This means that the model would automatically detect when it hits a plateau and can dynamically adjust the learning rate based on it.

12. Conclusion

This project aimed to create an image classification pipeline for the CIFAR-10 dataset using transfer learning together with a CNN algorithm. The final model achieved an accuracy of approximately 82%, a good result considering the multiple challenges faced by hardware limitations.

The process involved analyzing resources in search of a proper technique, choosing a pre-trained model to use for transfer learning, data preprocessing, the construction of a custom classification head for the MobileNetV2 base, model training, evaluations on performance and some visual analysis. Ultimately, the use of a more lightweight yet comprehensive pretrained model helped with the fine-tuning process in search of higher classification accuracy.

Overall, the project highlights the feasibility of transfer learning, especially when dealing with limited resources. It is worth noting that more powerful hardware, such as a higher-end GPU, would most likely produce better results. Enhanced computational resources means that the

model can make use of larger input sizes, the computer can sustain longer training durations and multiple fine-tuning attempts (Samuel, Sebe and Almeida, 2020).

References

DataSeries (n.d.) ‘Visualizing the Feature Maps and Filters by Convolutional Neural Networks’, *Medium*. Available at: https://medium.com/dataseries/visualizing-the-feature-maps-and-filters-by-convolutional-neural-networks-e1462340518e (Accessed: 14 July 2025).

Fernandes, A.F.A., Dórea, J.R.R. and Rosa, G.J. de M. (2020) ‘Image Analysis and Computer Vision Applications in Animal Sciences: An Overview’, *Frontiers in Veterinary Science* 7. Available at: https://doi.org/10.3389/fvets.2020.551269 (Accessed: 14 July 2025).

Hasan, M., Ullah, S., Khan, M.J. and Khurshid, K. (2019) ‘Comparative Analysis of SVM, ANN and CNN for Classifying Vegetation Species Using Hyperspectral Thermal Infrared Data’, *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W13, pp. 1861–1868. Available at: https://doi.org/10.5194/isprs-archives-xlii-2-w13-1861-2019 (Accessed: 14 July 2025).

Holbrook, R. (n.d.) ‘The Convolutional Classifier’, *Kaggle*. Available at: https://www.kaggle.com/code/ryanholbrook/the-convolutional-classifier (Accessed: 14 July 2025).

Kumar Shukla, R. and Kumar Tiwari, A. (2023) ‘Masked Face Recognition Using MobileNet V2 with Transfer Learning’, *Computer Systems Science and Engineering* 45(1), pp. 293–309. Available at: https://doi.org/10.32604/csse.2023.027986 (Accessed: 14 July 2025).

Kumar, T., Brennan, R., Mileo, A. and Bendechache, M. (2024) ‘Image Data Augmentation Approaches: A Comprehensive Survey and Future Directions’, *IEEE Access*. Available at: https://doi.org/10.1109/access.2024.3470122 (Accessed: 14 July 2025).

Salim, N.O.M. and Mohammed, A.K. (2024) ‘Comparative Analysis of Classical Machine Learning and Deep Learning Methods for Fruit Image Recognition and Classification’, *Traitement du Signal* 41(3), pp. 1331–1343. Available at: https://doi.org/10.18280/ts.410322 (Accessed: 14 July 2025).

Samuel, S., Sebe, N. and Almeida, J. (2020) ‘CNNs for JPEGs: A Study in Computational Cost’, *arXiv (Cornell)*. Available at: https://arxiv.org/abs/2012.14426 (Accessed: 14 July 2025).

Scratchpad (n.d.) ‘Notebookdb696730f2’, *Kaggle*. Available at: https://www.kaggle.com/code/scratchpad/notebookdb696730f2/edit (Accessed: 14 July 2025).

Simonyan, K. and Zisserman, A. (2015) ‘Very Deep Convolutional Networks for Large-Scale Image Recognition’, *arXiv(Cornell)*. Available at: https://arxiv.org/abs/1409.1556 (Accessed: 14 July 2025).

Vabalas, A., Gowen, E., Poliakoff, E. and Casson, A.J. (2019) ‘Machine Learning Algorithm Validation with a Limited Sample Size’, *PLOS ONE* 14(11), p. e0224365. Available at: https://doi.org/10.1371/journal.pone.0224365 (Accessed: 14 July 2025).