University of Essex

Online

ML_PCOM7E July 2025 B

Individual Presentation

Methodology

1. As a baseline preparatory step, the necessary Keras and related Python packages were loaded into a new Google Colab notebook.

2. The CIFAR-10 image recognition dataset was then downloaded into the notebook.

3. The pixel RGB values for each image were normalized to fall between 0 and 255 to conform to the standard usable range (Probst, Boulesteix and Bischl, 2019).

4. The category values for each image were one-hot encoded into binary vectors with a length of 10, corresponding to the 10 categories of images: 'bird', 'car', 'cat', 'deer', 'dog', 'frog', 'horse', 'plane', 'ship', and 'truck'.

5. The original split between 50,000 training images and 10,000 test images was retained.

II. Validation Set Rationale

Primary Reasons

1. Training sets must be separated from validation sets to avoid overfitting a model (Burkov, 2019; Côté et al, 2024).

2. Training sets must be separated from validation sets to provide unknown (to the trained model) data points that are able to act as appropriate objective measures of the model's accuracy when faced with new information (Probst, Boulesteix and Bischl, 2019).

3. Training sets must be separated from validation sets to reduce the chance of introducing inadvertent human bias or data cherry-picking into the model (Burkov, 2019; Côté et al, 2024).

4. Validation sets allow the use of iterative adjustments to hyperparameters to reach the most efficient or most optimized version of the model (Burkov, 2019; Côté et al, 2024).

Convolutional Neural Network ("CNN")

1. Industry standard with robust code packages available.

2. Well-understood in the literature (Burkov, 2019; Côté et al, 2024).

3. Readily amenable to hyperparameter adjustments during the training and validation steps (Burkov, 2019; Côté et al, 2024).

4. Readily amenable to the inclusion of optimization and related processing layers in the code (Probst, Boulesteix and Bischl, 2019).

Keras 2D CNN Architecture Components

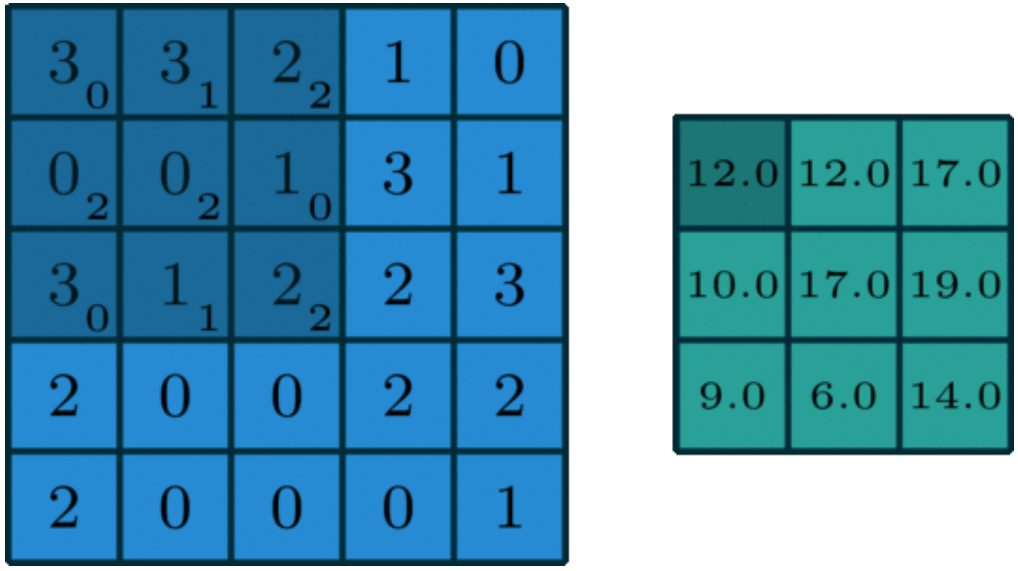
1. The core component is a kernel, or a matrix of weights (Burkov, 2019; Côté et al, 2024).

2. The kernel passes over matrices of input data sequentially (Burkov, 2019; Côté et al, 2024; *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch*).

3. The kernel performs element-based multiplication using the values of the data "window overlap" it occupies at each point (Burkov, 2019; Côté et al, 2024; *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch*).

4. The results of that multiplication are assigned to a single output value (Burkov, 2019; Côté et al, 2024; *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch*).

Example of a CNN Kernel in Action



Source: *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch.*

Relevant Code

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=x_train.shape[1:]))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='softmax'))
```

Adapted From: *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch.*

Initial Hyperparameters

1. Epochs: 5
2. Batch size: 64
3. Optimizer function: Adam
4. Loss function: Categorical cross-entropy
5. Final layer activation function: Softmax
6. Please note that the initial hyperparameters were adapted from *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch.*

Methodology

1. The model was trained eight (8) times using variated hyperparameters.

2. The first run used a batch of 5 epochs, followed by batches of 25, then 50 epochs. Each used an 0.5 dropout rate and the Keras Adam optimizer.

3. Based on the results of the preceding, a batch of 25 epochs was run at an 0.25 dropout rate, also using the Keras Adam optimizer.

4. For comparison, batches of 25 epochs were run at 0.5 and 0.25 dropout rates using the Keras AdamW optimizer.

5. For further comparison, batches of 25 epochs were run at 0.5 and 0.25 dropout rates using the Stochastic Gradient Descent (”SGD”) optimizer.

VI. Performance Metrics

Result: 5 Epochs @ 0.5 Dropout Rate + Adam Optimizer

1. After 5 epochs, the model's accuracy reached 69.07%, which far exceeded the accuracy of pure chance (10.00%, or guessing 1 out of the 10 available image categories).

```
Epoch 1/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 76s 94ms/step - accuracy: 0.3034 - loss: 1.8685 - val_accuracy: 0.5320 - val_loss: 1.2893
Epoch 2/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.5172 - loss: 1.3608 - val_accuracy: 0.6083 - val_loss: 1.1125
Epoch 3/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 84s 95ms/step - accuracy: 0.5870 - loss: 1.1706 - val_accuracy: 0.6261 - val_loss: 1.0595
Epoch 4/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.6287 - loss: 1.0548 - val_accuracy: 0.6644 - val_loss: 0.9574
Epoch 5/5
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 91ms/step - accuracy: 0.6661 - loss: 0.9655 - val_accuracy: 0.6907 - val_loss: 0.8769
Test Loss: 0.8769
Test Accuracy: 0.6907
```

VI. Performance Metrics

Result: 25 Epochs @ 0.5 Dropout Rate + Adam Optimizer

1. After 25 epochs, the model's accuracy reached 74.19%, which exceeded the accuracy of 5 epochs of training (69.07%) and of pure chance (10.00%).

```
Epoch 21/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 71s 91ms/step - accuracy: 0.8445 - loss: 0.4334 - val_accuracy: 0.7351 - val_loss: 0.9080
Epoch 22/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 74s 94ms/step - accuracy: 0.8515 - loss: 0.4105 - val_accuracy: 0.7375 - val_loss: 0.9336
Epoch 23/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.8580 - loss: 0.3955 - val_accuracy: 0.7296 - val_loss: 0.9713
Epoch 24/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 72s 92ms/step - accuracy: 0.8626 - loss: 0.3807 - val_accuracy: 0.7423 - val_loss: 0.9861
Epoch 25/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 91ms/step - accuracy: 0.8679 - loss: 0.3579 - val_accuracy: 0.7419 - val_loss: 1.0186
Test Loss: 1.0186
Test Accuracy: 0.7419
```

VI. Performance Metrics

Result: 50 Epochs @ 0.5 Dropout Rate + Adam Optimizer

1. After 50 epochs, the model's accuracy logged in at 72.45%, which exceeded 5 epochs of training (69.07%), but was less accurate compared to 25 epochs of training (74.19%).

2. Additional epochs beyond approximately 25 were therefore determined to be likely to provide no material increase in accuracy.

```
Epoch 46/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 92ms/step - accuracy: 0.9047 - loss: 0.2507 - val_accuracy: 0.7312 - val_loss: 1.4158
Epoch 47/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 78s 99ms/step - accuracy: 0.9139 - loss: 0.2260 - val_accuracy: 0.7309 - val_loss: 1.4279
Epoch 48/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 79s 101ms/step - accuracy: 0.9041 - loss: 0.2562 - val_accuracy: 0.7310 - val_loss: 1.3926
Epoch 49/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 102ms/step - accuracy: 0.9105 - loss: 0.2447 - val_accuracy: 0.7302 - val_loss: 1.4579
Epoch 50/50
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 104ms/step - accuracy: 0.9168 - loss: 0.2257 - val_accuracy: 0.7245 - val_loss: 1.4063
Test Loss: 1.4063
Test Accuracy: 0.7245
```

VI. Performance Metrics

Result: 25 Epochs @ 0.25 Dropout Rate + Adam Optimizer

1. Model accuracy reached 72.18%, comparable to previous runs.

2. From this result, the conclusion is reached that further runs with variations on the hyperparameters used are unlikely to exceed the accuracy measurements reached to this point.

```
Epoch 21/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 92ms/step - accuracy: 0.9108 - loss: 0.2423 - val_accuracy: 0.7305 - val_loss: 1.1705
Epoch 22/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 72s 92ms/step - accuracy: 0.9161 - loss: 0.2285 - val_accuracy: 0.7304 - val_loss: 1.1425
Epoch 23/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 72s 93ms/step - accuracy: 0.9210 - loss: 0.2145 - val_accuracy: 0.7235 - val_loss: 1.2117
Epoch 24/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 72s 93ms/step - accuracy: 0.9227 - loss: 0.2114 - val_accuracy: 0.7160 - val_loss: 1.2361
Epoch 25/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.9252 - loss: 0.2033 - val_accuracy: 0.7218 - val_loss: 1.3159
Test Loss: 1.3159
Test Accuracy: 0.7218
```

VI. Performance Metrics

Result: 25 Epochs @ 0.50 Dropout Rate + AdamW Optimizer

1. Switching to the AdamW optimizer resulted in a comparable accuracy of prior runs at 73.22%.

2. At first glance, the AdamW optimizer did not improve the model.

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 102ms/step - accuracy: 0.8587 - loss: 0.3910 - val_accuracy: 0.7400 - val_loss: 0.9261
Epoch 21/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 84s 107ms/step - accuracy: 0.8581 - loss: 0.3851 - val_accuracy: 0.7345 - val_loss: 0.9681
Epoch 22/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 103ms/step - accuracy: 0.8677 - loss: 0.3648 - val_accuracy: 0.7276 - val_loss: 1.0097
Epoch 23/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 104ms/step - accuracy: 0.8692 - loss: 0.3549 - val_accuracy: 0.7356 - val_loss: 1.0354
Epoch 24/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 105ms/step - accuracy: 0.8806 - loss: 0.3320 - val_accuracy: 0.7423 - val_loss: 0.9946
Epoch 25/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 103ms/step - accuracy: 0.8803 - loss: 0.3262 - val_accuracy: 0.7322 - val_loss: 1.0374
Test Loss: 1.0374
Test Accuracy: 0.7322
```

VI. Performance Metrics

Result: 25 Epochs @ 0.25 Dropout Rate + AdamW Optimizer

1. To confirm, adjusting the dropout rate to 0.25 resulted in a minor decrease in accuracy to 72.10%.

2. The conclusion must be had that for this dataset, there is no significant difference between the Adam and AdamW optimizers to model accuracy.

```
782/782 ━━━━━━━━━━━━━━━━━━━━ 80s 100ms/step - accuracy: 0.8916 - loss: 0.3006 - val_accuracy: 0.7125 - val_loss: 1.1125
Epoch 21/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 84s 103ms/step - accuracy: 0.8965 - loss: 0.2794 - val_accuracy: 0.7130 - val_loss: 1.1113
Epoch 22/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 78s 100ms/step - accuracy: 0.9027 - loss: 0.2660 - val_accuracy: 0.7207 - val_loss: 1.1914
Epoch 23/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 101ms/step - accuracy: 0.9019 - loss: 0.2660 - val_accuracy: 0.7136 - val_loss: 1.2365
Epoch 24/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 75s 96ms/step - accuracy: 0.9140 - loss: 0.2331 - val_accuracy: 0.7144 - val_loss: 1.2571
Epoch 25/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 86s 102ms/step - accuracy: 0.9124 - loss: 0.2328 - val_accuracy: 0.7210 - val_loss: 1.2490
Test Loss: 1.2490
Test Accuracy: 0.7210
```

VI. Performance Metrics

Result: 25 Epochs @ 0.50 Dropout Rate + SGD Optimizer

1. Switching to the Stochastic Gradient Descent (“SGD”) optimizer resulted in a comparable validation accuracy (73.89%).

2. The SGD optimizer was chosen as it is computationally efficient for larger datasets and its use of smaller batches reduces the risk of outlier points / local minima / saddle points that could skew the results.

```
Epoch 21/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 93ms/step - accuracy: 0.8033 - loss: 0.5504 - val_accuracy: 0.7206 - val_loss: 0.8476
Epoch 22/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 93ms/step - accuracy: 0.8155 - loss: 0.5197 - val_accuracy: 0.7345 - val_loss: 0.8321
Epoch 23/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 95ms/step - accuracy: 0.8218 - loss: 0.4982 - val_accuracy: 0.7356 - val_loss: 0.8211
Epoch 24/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.8327 - loss: 0.4718 - val_accuracy: 0.7360 - val_loss: 0.8265
Epoch 25/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 81s 92ms/step - accuracy: 0.8352 - loss: 0.4549 - val_accuracy: 0.7389 - val_loss: 0.8354
Test Loss: 0.8354
Test Accuracy: 0.7389
```

VI. Performance Metrics

Result: 25 Epochs @ 0.25 Dropout Rate + SGD Optimizer

1. Switching to the Stochastic Gradient Descent ("SGD") optimizer resulted in a materially similar accuracy score of 72.27%.

2. After training the model on 3 different optimizers, each returned markedly similar accuracy results. The similarity of outcomes suggests that the dataset itself might not be amenable to CNN model accuracy values better than the values already found.

```
Epoch 21/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 83s 95ms/step - accuracy: 0.8554 - loss: 0.4030 - val_accuracy: 0.7210 - val_loss: 0.9259
Epoch 22/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 82s 95ms/step - accuracy: 0.8665 - loss: 0.3726 - val_accuracy: 0.7171 - val_loss: 0.9381
Epoch 23/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 71s 91ms/step - accuracy: 0.8757 - loss: 0.3502 - val_accuracy: 0.7087 - val_loss: 1.0033
Epoch 24/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.8847 - loss: 0.3186 - val_accuracy: 0.7107 - val_loss: 1.0334
Epoch 25/25
782/782 ━━━━━━━━━━━━━━━━━━━━ 73s 93ms/step - accuracy: 0.8913 - loss: 0.3064 - val_accuracy: 0.7227 - val_loss: 1.0235
Test Loss: 1.0235
Test Accuracy: 0.7227
```

VII. Comparative Discussion

Hyperparameters	Validation Accuracy
5 Epochs @ 0.5 Dropout Rate + Adam Optimizer	69.07%
25 Epochs @ 0.5 Dropout Rate + Adam Optimizer	**74.19%**
50 Epochs @ 0.5 Dropout Rate + Adam Optimizer	72.45%
25 Epochs @ 0.25 Dropout Rate + Adam Optimizer	72.18%
25 Epochs @ 0.50 Dropout Rate + AdamW Optimizer	73.22%
25 Epochs @ 0.25 Dropout Rate + AdamW Optimizer	72.10%
25 Epochs @ 0.50 Dropout Rate + SGD Optimizer	73.89%
25 Epochs @ 0.25 Dropout Rate + SGD Optimizer	72.27%

VII. Comparative Discussion

Results

1. Increasing the number of epochs increased accuracy, with the greatest accuracy after 25 epochs, but decreased accuracy at 50 epochs.

2. Decreasing the dropout rate from 0.5 to 0.25 provided no particularly significant improvement to model accuracy.

3. Using the Keras AdamW optimizer, rather than the Keras Adam optimizer, likewiseprovided no particularly significant improvement to model accuracy.

4. Using the Keras Stochastic Gradient Descent optimizer provided no particularly significant improvement to model accuracy.

Lessons Learned

1. Hyperparameter tuning may or may not increase model accuracy, but more is not always better in terms of hyperparameter tuning (Probst, Boulesteix and Bischl, 2019).

2. Model accuracy improvements are not necessarily always linear. Changing some hyperparameters, such as the epoch batch size, may result in diminishing returns or decreased model accuracy after an inflection point (Burkov, 2019; Côté et al, 2024).

3. Likewise, hyperparameter interactions cannot be assumed to follow 1-to-1 linear relationships between each other, either, but may interact non-linearly to increase or reduce model accuracy (Arnold et al, 2024).

4. Optimizer functions may or may not significantly impact model accuracy. Experimenting with several is worthwhile when fine-tuning a model, either to reach increased accuracy, or to confirm the utility of prior tests (Probst, Boulesteix and Bischl, 2019).

References

Arnold, C., Biedebach, L., Küpfer, A. and Neunhoeffer, M. (2024). The role of hyperparameters in machine learning models and how to tune them. *Political Science Research and Methods*, [online] pp.1–8. doi:https://doi.org/10.1017/ psrm.2023.61.

Burkov, A. (2019). *The Hundred-Page Machine Learning Book*. Andriy Burkov.
Côté, P.-O., Nikanjam, A., Ahmed, N., Humeniuk, D. and Khomh, F. (2024). Data cleaning and machine learning: a systematic literature review. *Automated software engineering*, 31(2). doi:https://doi.org/10.1007/s10515-024-00453-w.

colab.research.google.com (2025). *Google Colaboratory Notebook: CIFAR 10-CNN Using PyTorch*. [online] Available at: https://colab.research.google.com.

References (Continued)

Côté, P.-O., Nikanjam, A., Ahmed, N., Humeniuk, D. and Khomh, F. (2024). Data cleaning and machine learning: a systematic literature review. *Automated software engineering*, 31(2). doi:https://doi.org/10.1007/s10515-024-00453-w.

kaggle.com (2025). *CIFAR 10-CNN Using PyTorch*. [online] Available at: https://www.kaggle.com/code/shadabhussain/cifar-10-cnn-using-pytorch.

Probst, P., Boulesteix, A.-L. and Bischl, B. (2019). Tunability: Importance of Hyperparameters of Machine Learning Algorithms. *Journal of Machine Learning Research*, [online] 20(53), pp.1–32. Available at: https://www.jmlr.org/papers/v20/18-444.html.