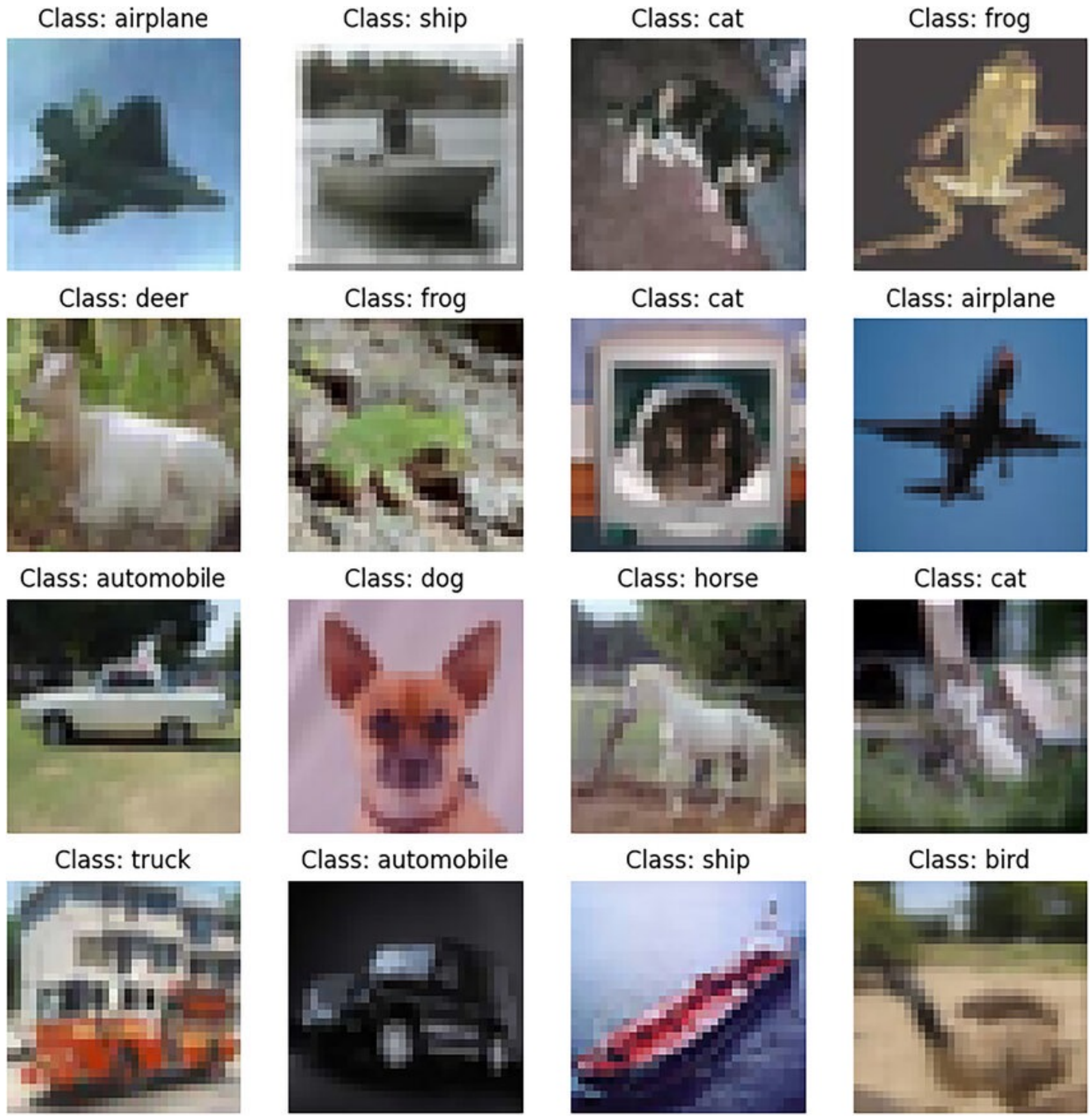
Object Recognition with CNNs & Transfer Learning (CIFAR-10)

https://abcnews.go.com/Lifestyle/animal-food-meme-sweeping-internet-leave-stumped/story?id=37554574



https://www.researchgate.net/publication/391119246_Enhancing_CNNs_via_structural_intervention
_with_XGBoost?_tp=eyJjb250ZXh0Ijp7ImZpcnN0UGFnZSI6Il9kaXJlY3QiLCJwYWdlIjoiX2RpcmVjdCJ9fQ

Dataset & Pre-processing

CIFAR-10: 60k images, 10 classes, 32×32×3

Train/Val/Test split: 45k/5k/10k

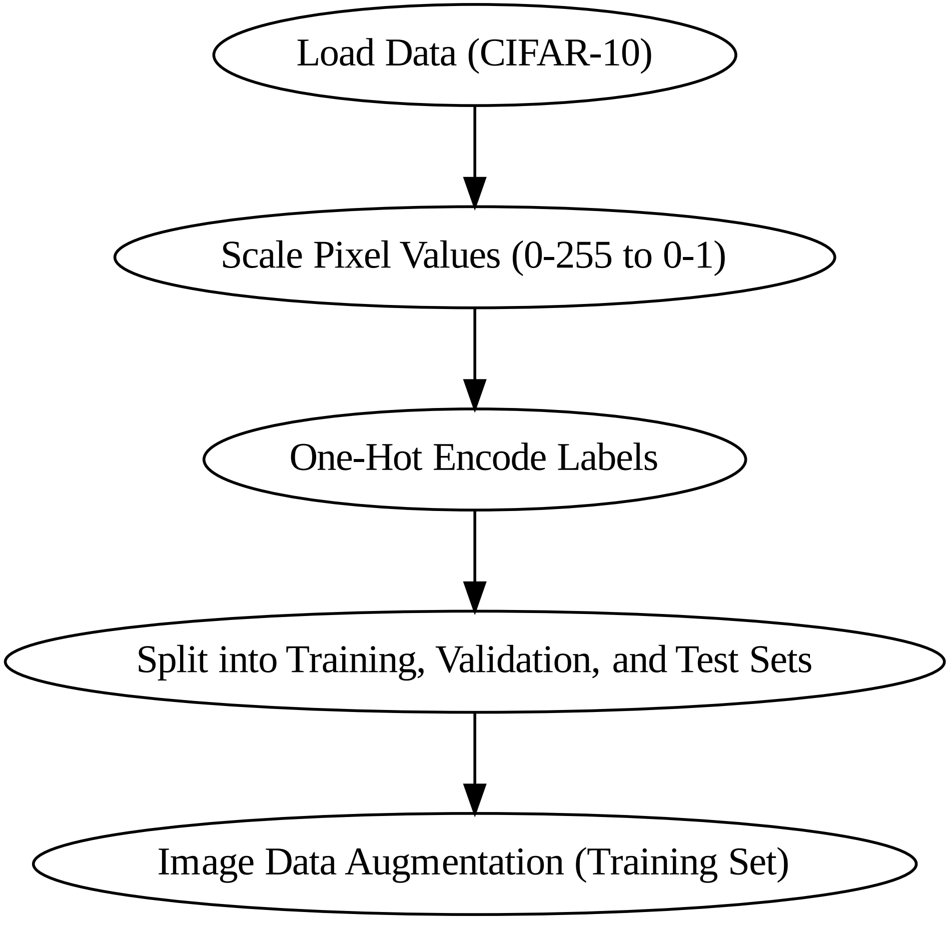
Normalisation, one-hot encoding, consistent preprocessing

Data Augmentation

Random flip, rotation, zoom, translation

Enhances diversity and generalisation

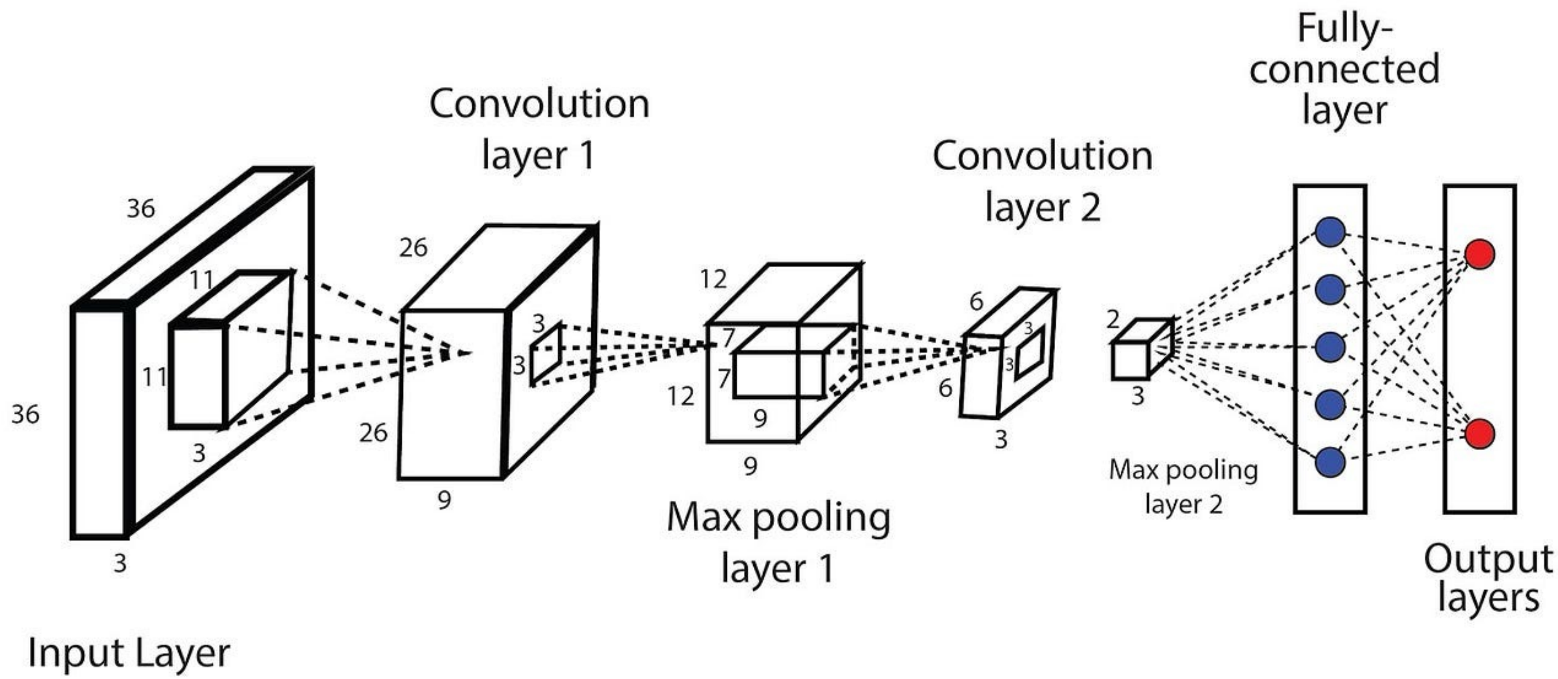
Applied during training to both models


Load Data (CIFAR-10)
Scale Pixel Values (0-255 to 0-1)
One-Hot Encode Labels
Split into Training, Validation, and Test Sets
Image Data Augmentation (Training Set)

Model 1 – Baseline CNN (from scratch)

Conv–Conv–Pool ×2 → Conv → Dense(256) → Dropout → Softmax(10)

Optimizer: Adam (1e-3), Loss: CCE, Dropout regularisation



https://medium.com/analytics-vidhya/classifying-cifar-10-using-a-simple-cnn-4e9a6dd7600b

--- Custom CNN Model Summary (without data augmentation) ---

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 29, 29, 32)	1,568
max_pooling2d_2 (MaxPooling2D)	(None, 14, 14, 32)	0
conv2d_3 (Conv2D)	(None, 11, 11, 32)	16,416
max_pooling2d_3 (MaxPooling2D)	(None, 5, 5, 32)	0
flatten_2 (Flatten)	(None, 800)	0
dense_4 (Dense)	(None, 256)	205,056
dense_5 (Dense)	(None, 10)	2,570

Total params: 676,832 (2.58 MB)
Trainable params: 225,610 (881.29 KB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 451,222 (1.72 MB)

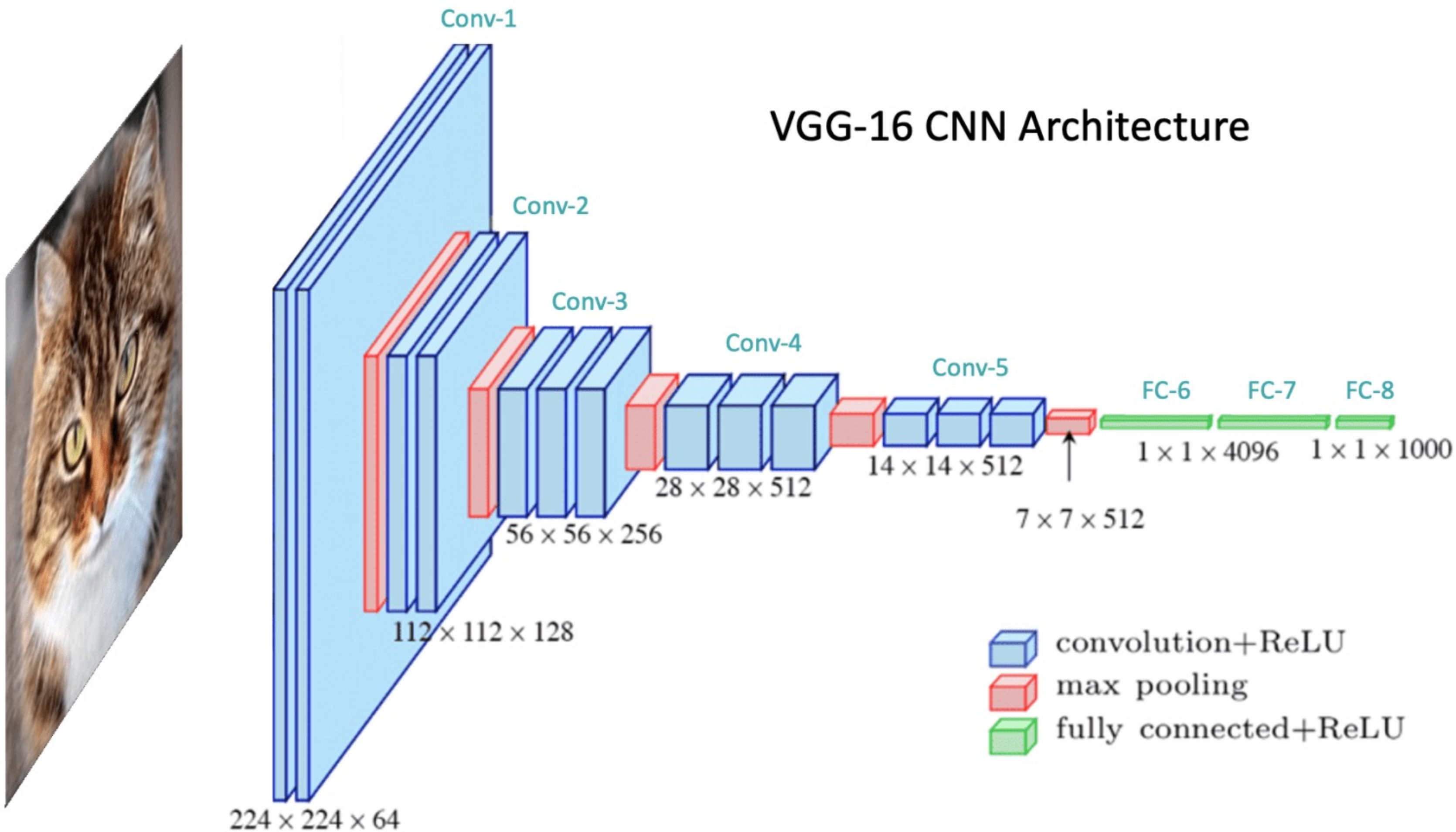
Model 2 – Transfer Learning (VGG16 fine-tune)

Base: VGG16 pretrained on ImageNet, include_top=False

Freeze early layers; fine-tune last block

Custom head: GAP → Dense(256) → Dropout → Softmax(10)

VGG-16 CNN Architecture



https://learnopencv.com/understanding-convolutional-neural-networks-cnn/

--- VGG16 Model Summary ---

Model: "sequential_3"

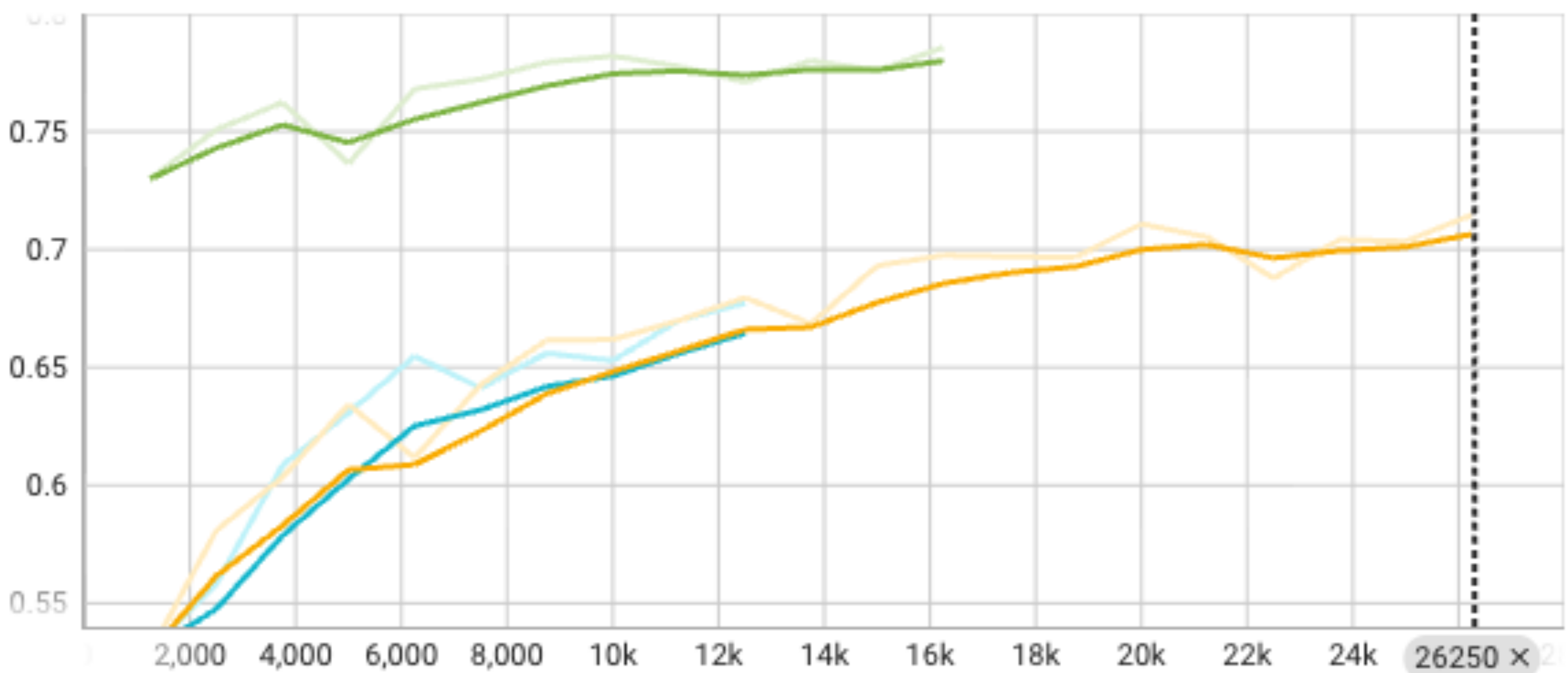
Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 3, 3, 512)	14,714,688
flatten_3 (Flatten)	(None, 4608)	0
dense_6 (Dense)	(None, 256)	1,179,904
dense_7 (Dense)	(None, 10)	2,570

Total params: 18,262,112 (69.66 MB)
Trainable params: 1,182,474 (4.51 MB)
Non-trainable params: 14,714,688 (56.13 MB)
Optimizer params: 2,364,950 (9.02 MB)

Training Curves (Accuracy/Loss)

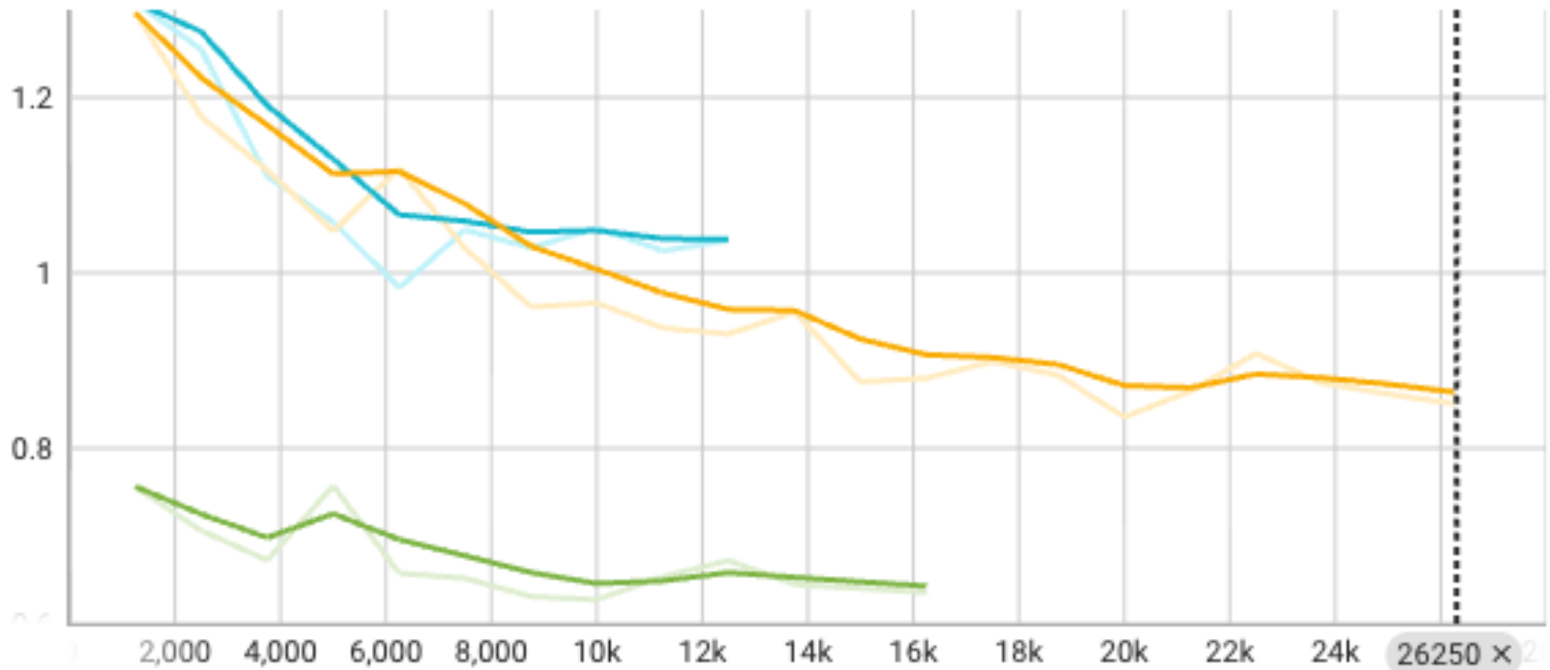
Compare CNN vs VGG16 training & validation accuracy

VGG16 converges faster; CNN may overfit earlier evaluation_accuracy_vs_iterations



	Run ↑	Smoothed	Value	Step	Relative
●	cnn_aug/validation	0.7065	0.7146	26,250	7.766 min
●	cnn_no_aug/validation	0.6645	0.6773	12,500	39.85 sec
●	vgg16/validation	0.7799	0.7856	16,250	17.45 min

evaluation_loss_vs_iterations



	Run ↑	Smoothed	Value	Step	Relative
	cnn_aug/validation	0.864	0.8507	26,250	7.766 min
	cnn_no_aug/validation	1.0381	1.0369	12,500	39.85 sec
	vgg16/validation	0.6428	0.6355	16,250	17.45 min

Results & Confusion Matrices

Validation/Test Accuracy for both models

Confusion matrices highlight misclassifications

VGG16 reduces class confusion vs CNN

Model	Data Augmentation	Test Loss	Test Accuracy
Custom CNN	No	1.0458	67.20%
Custom CNN	Yes	1.0458	70.78%
VGG16	Yes	0.6369	78.03%

Unit09 Ex1 Convolutional Neural Networks (CNN) - Object Recognition.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

Share

RAM Disk

```python
except Exception as e:
    print(f"Error loading model from {model_aug_path}: {e}")
    loaded_model_aug = None # Set to None if loading fails

try:
    loaded_model_vgg16 = tf.keras.models.load_model(model_vgg16_path)
    print(f"Model loaded successfully from: {model_vgg16_path}")
except Exception as e:
    print(f"Error loading model from {model_vgg16_path}: {e}")
    loaded_model_vgg16 = None # Set to None if loading fails
```

```
Mounted at /content/drive
Model loaded successfully from: /content/drive/MyDrive/saved_models/cnn_no_aug.keras
Model loaded successfully from: /content/drive/MyDrive/saved_models/cnn_aug.keras
Model loaded successfully from: /content/drive/MyDrive/saved_models/vgg16_cifar10.keras
```

[11] 0s

```python
# Evaluate Custom CNN without Data Augmentation
print("\nEvaluating Custom CNN without Data Augmentation:")
# Assuming 'model_cnn' was trained without augmentation in the last run of cell 0ce4775e
loss_aug, accuracy_aug = loaded_model_no_aug.evaluate(x_test, y_cat_test, verbose=0)
print(f"Test Loss: {loss_aug:.4f}")
print(f"Test Accuracy: {accuracy_aug:.4f}")
```

```
Evaluating Custom CNN without Data Augmentation:
Test Loss: 1.0458
Test Accuracy: 0.6720
```

[ ]

```python
# Evaluate Custom CNN with Data Augmentation
print("Evaluating Custom CNN with Data Augmentation:")
loss_no_aug, accuracy_aug = loaded_model_aug.evaluate(x_test, y_cat_test, verbose=0)
print(f"Test Loss: {loss_aug:.4f}")
print(f"Test Accuracy: {accuracy_aug:.4f}")
```

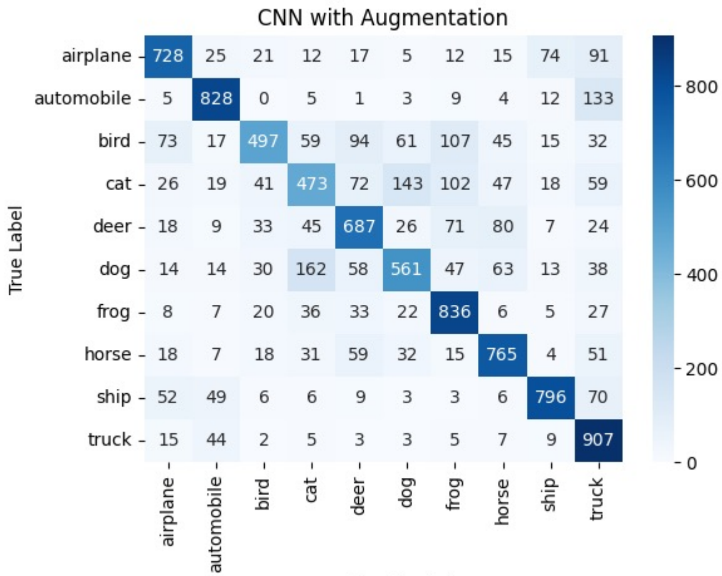
```
Evaluating Custom CNN with Data Augmentation:
Test Loss: 1.0458
Test Accuracy: 0.7078
```

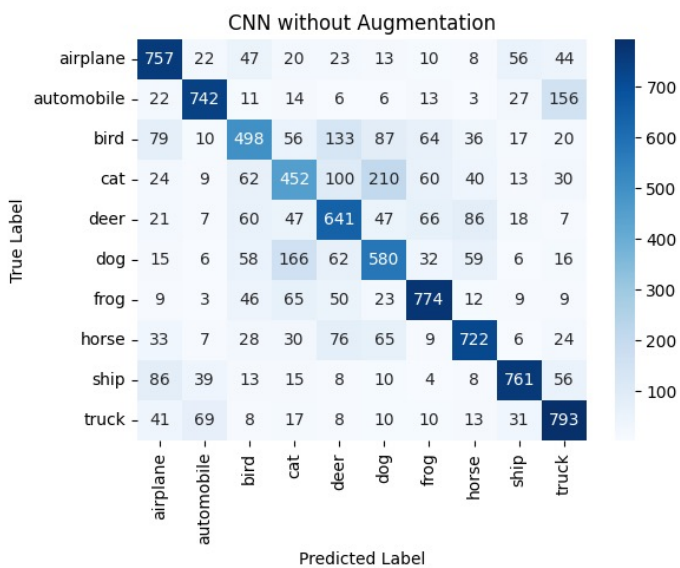
[ ]

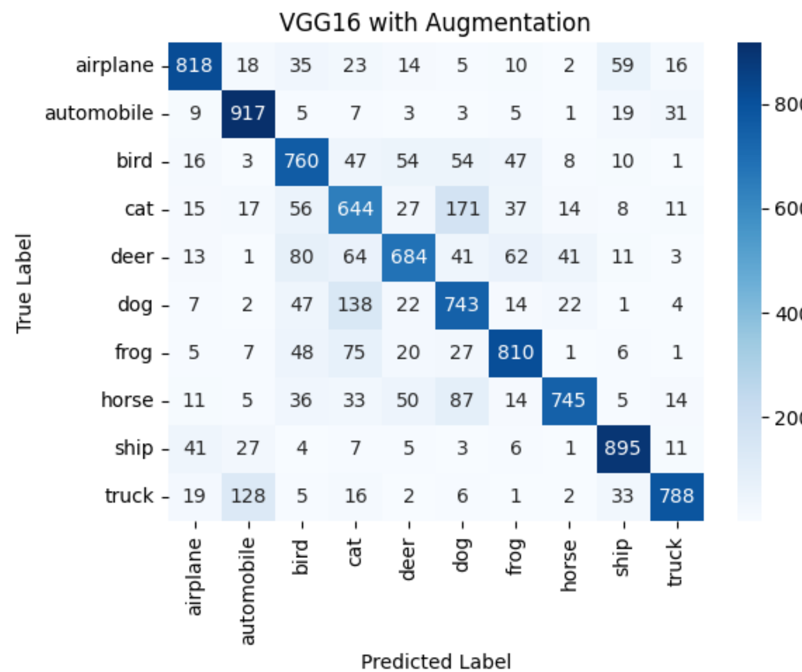
```python
# Evaluate VGG16 with Data Augmentation
print("\nEvaluating VGG16 with Data Augmentation:")
# Assuming 'model_vgg16' was trained with augmentation in the last run of cell 8dfb978d
loss_vgg16, accuracy_vgg16 = loaded_model_vgg16.evaluate(x_test_resized, y_cat_test_vgg16, verbose=0)
print(f"Test Loss: {loss_vgg16:.4f}")
print(f"Test Accuracy: {accuracy_vgg16:.4f}")
```

```
Evaluating VGG16 with Data Augmentation:
Test Loss: 0.6369
Test Accuracy: 0.7803
```

Double-click (or enter) to edit


CNN with Augmentation
True Label
Predicted Label
airplane 728 25 21 12 17 5 12 15 74 91
automobile 5 828 0 5 1 3 9 4 12 133
bird 73 17 497 59 94 61 107 45 15 32
cat 26 19 41 473 72 143 102 47 18 59
deer 18 9 33 45 687 26 71 80 7 24
dog 14 14 30 162 58 561 47 63 13 38
frog 8 7 20 36 33 22 836 6 5 27
horse 18 7 18 31 59 32 15 765 4 51
ship 52 49 6 6 9 3 3 6 796 70
truck 15 44 2 5 3 3 5 7 9 907
airplane automobile bird cat deer dog frog horse ship truck
0 200 400 600 800


CNN without Augmentation
True Label
Predicted Label
airplane 757 22 47 20 23 13 10 8 56 44
automobile 22 742 11 14 6 6 13 3 27 156
bird 79 10 498 56 133 87 64 36 17 20
cat 24 9 62 452 100 210 60 40 13 30
deer 21 7 60 47 641 47 66 86 18 7
dog 15 6 58 166 62 580 32 59 6 16
frog 9 3 46 65 50 23 774 12 9 9
horse 33 7 28 30 76 65 9 722 6 24
ship 86 39 13 15 8 10 4 8 761 56
truck 41 69 8 17 8 10 10 13 31 793
airplane automobile bird cat deer dog frog horse ship truck
100 200 300 400 500 600 700


VGG16 with Augmentation
True Label
Predicted Label
airplane 818 18 35 23 14 5 10 2 59 16
automobile 9 917 5 7 3 3 5 1 19 31
bird 16 3 760 47 54 54 47 8 10 1
cat 15 17 56 644 27 171 37 14 8 11
deer 13 1 80 64 684 41 62 41 11 3
dog 7 2 47 138 22 743 14 22 1 4
frog 5 7 48 75 20 27 810 1 6 1
horse 11 5 36 33 50 87 14 745 5 14
ship 41 27 4 7 5 3 6 1 895 11
truck 19 128 5 16 2 6 1 2 33 788
airplane automobile bird cat deer dog frog horse ship truck
200 400 600 800

CNN no augmentation

Class	Precision	Recall	F1-Score	Support
airplane	0.7	0.76	0.73	1000
automobile	0.81	0.74	0.78	1000
bird	0.6	0.5	0.54	1000
cat	0.51	0.45	0.48	1000
deer	0.58	0.64	0.61	1000
dog	0.55	0.58	0.57	1000
frog	0.74	0.77	0.76	1000
horse	0.73	0.72	0.73	1000
ship	0.81	0.76	0.78	1000
truck	0.69	0.79	0.74	1000
Accuracy	**nan**	**nan**	**0.67**	**10000**
Macro avg	**0.67**	**0.67**	**0.67**	**10000**
Weighted avg	**0.67**	**0.67**	**0.67**	**10000**

CNN with augmentation

Class	Precision	Recall	F1-Score	Support
airplane	0.76	0.73	0.74	1000
automobile	0.81	0.83	0.82	1000
bird	0.74	0.5	0.6	1000
cat	0.57	0.47	0.52	1000
deer	0.67	0.69	0.68	1000
dog	0.65	0.56	0.6	1000
frog	0.69	0.84	0.76	1000
horse	0.74	0.77	0.75	1000
ship	0.84	0.8	0.82	1000
truck	0.63	0.91	0.75	1000
Accuracy	**nan**	**nan**	**0.71**	**10000**
Macro avg	**0.71**	**0.71**	**0.7**	**10000**
Weighted avg	**0.71**	**0.71**	**0.7**	**10000**

VGG16

Class	Precision	Recall	F1-Score	Support
airplane	0.86	0.82	0.84	1000
automobile	0.82	0.92	0.86	1000
bird	0.71	0.76	0.73	1000
cat	0.61	0.64	0.63	1000
deer	0.78	0.68	0.73	1000
dog	0.65	0.74	0.69	1000
frog	0.81	0.81	0.81	1000
horse	0.89	0.74	0.81	1000
ship	0.85	0.9	0.87	1000
truck	0.9	0.79	0.84	1000
Accuracy	**nan**	**nan**	**0.78**	**10000**
Macro avg	**0.79**	**0.78**	**0.78**	**10000**
Weighted avg	**0.79**	**0.78**	**0.78**	**10000**

Discussion – Strengths & Trade-offs

CNN: lightweight, flexible, lower compute

Transfer Learning: higher accuracy, faster convergence, heavier mode

Lessons Learned & Insights

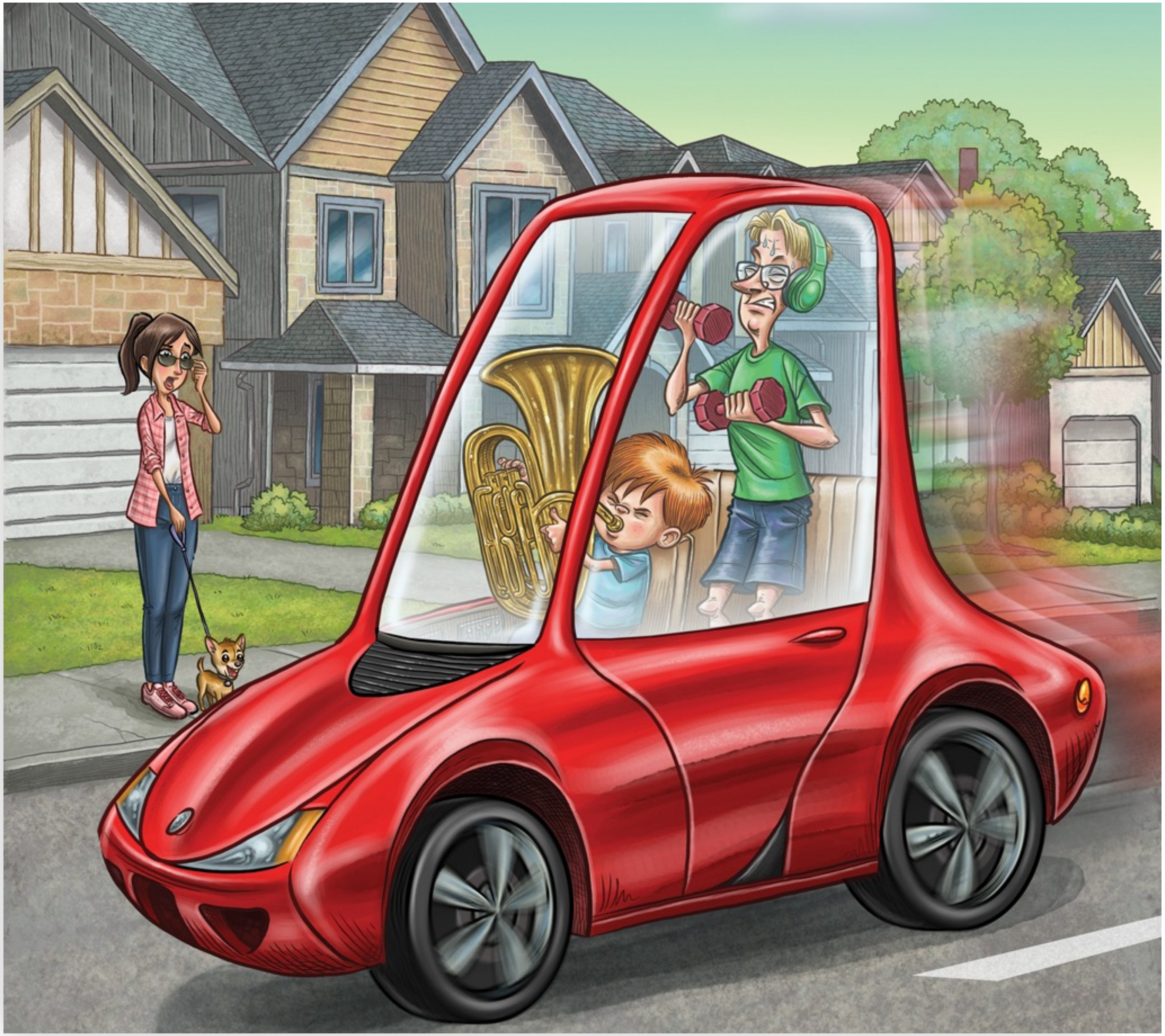
- Hyperparameter tuning
- Importance of Validation Set
- Power of data-augmentation

Applications, Ethics & Future Work

Autonomous Vehicles, Biometric Authentication, Medical Imaging

Ethical and practical dimensions of AI

ResNet50/MobileNetV2



https://stanfordmag.org/contents/in-two-years-there-could-be-10-million-self-driving-cars-on-the-roads

References

Bender, E.M., Gebru, T., McMillan-Major, A. and Shmitchell, S. (2021) *On the dangers of stochastic parrots: Can language models be too big?* In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21).* New York: ACM Press.

Esteva, A., et al. (2019) *A guide to deep learning in healthcare. Nature Medicine*, 25, pp.24–29.

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep learning.* Cambridge, MA: MIT Press.

Hien, B. M. (2023) *Solving CIFAR-10 dataset with VGG16 pre-trained architect using PyTorch.* Available at: https://medium.com/@buiminhhien2k/solving-cifar10-dataset-with-vgg16-pre-trained-architect-using-pytorch-validation-accuracy-over-3f9596942861.

Howard, J. and Ruder, S. (2018) *Universal language model fine-tuning for text classification.* In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL).* Melbourne: ACL, pp.328–339.

Jobin, A., Ienca, M. and Vayena, E. (2019) *The global landscape of AI ethics guidelines. Nature Machine Intelligence*, 1(9), pp.389–399.

Kingma, D.P. and Ba, J. (2015) *Adam: A method for stochastic optimization.* In: *Proceedings of the International Conference on Learning Representations (ICLR 2015).* Available at: https://arxiv.org/abs/1412.6980

Krizhevsky, A., Nair, V. and Hinton, G. (2009) *The CIFAR-10 dataset.* Toronto: University of Toronto. Available at: https://www.cs.toronto.edu/~kriz/cifar.html

References Continued

LeCun, Y., Bengio, Y. and Hinton, G. (2015) *Deep learning. Nature*, 521(7553), pp.436–444.

LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998) *Gradient-based learning applied to document recognition. Proceedings of the IEEE*, 86(11), pp.2278–2324.

Shorten, C. and Khoshgoftaar, T.M. (2019) *A survey on image data augmentation for deep learning. Journal of Big Data*, 6(1), 60.

Simonyan, K. and Zisserman, A. (2015) *Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICLR 2015).* Available at: https://arxiv.org/abs/1409.1556

Yosinski, J., Clune, J., Bengio, Y. and Lipson, H. (2014) *How transferable are features in deep neural networks?* In: *Proceedings of the 27th International Conference on Neural Information Processing Systems (NeurIPS 2014).* Montréal, Canada.

Zhang, Y., Bengio, S., Hardt, M., Recht, B. and Vinyals, O. (2021) *Understanding deep learning requires rethinking generalization. International Conference on Learning Representations (ICLR 2021).* Available at: https://arxiv.org/abs/1611.03530