Computer Vision Assignment

Hugo Collins

40446925

Dataset Preparation

```
Folder: n02808440 - Label: 0 - Class Name: bathtub, bathing tub, bath, tub
Folder: n01742172 - Label: 1 - Class Name: boa constrictor, Constrictor constrictor
Folder: n01443537 - Label: 2 - Class Name: goldfish, Carassius auratus
Folder: n02909870 - Label: 3 - Class Name: bucket, pail
Folder: n02481823 - Label: 4 - Class Name: chimpanzee, chimp, Pan troglodytes
Folder: n02364673 - Label: 5 - Class Name: guinea pig, Cavia cobaya
Folder: n02788148 - Label: 6 - Class Name: bannister, banister, balustrade, balusters, handrail
Folder: n01984695 - Label: 7 - Class Name: spiny lobster, langouste, rock lobster, crawfish, crayfish, sea crawfish
Folder: n02504458 - Label: 8 - Class Name: African elephant, Loxodonta africana
Folder: n03126707 - Label: 9 - Class Name: crane
Folder: n03179701 - Label: 10 - Class Name: desk
Folder: n02509815 - Label: 11 - Class Name: lesser panda, red panda, panda, bear cat, cat bear, Ailurus fulgens
Folder: n02730930 - Label: 12 - Class Name: apron
Folder: n03160309 - Label: 13 - Class Name: dam, dike, dyke
Folder: n02769748 - Label: 14 - Class Name: backpack, back pack, knapsack, packsack, rucksack, haversack
```

- First 400 from each folder selected for training
- Last 100 selected for testing
- For fine tuning I needed a validation set
- So split 400 training into 300/100 with the last 100 used for validation
- Final split per folder
- 300/100/100
- Final Split Overall
- 4500 Training /1500 Validation /1500

Handcrafted Features

Preprocessing Differences:

SIFT – Used histogram equalisation and masking to enhance image

ORB – Resized images to 256 x 256

Model	Num_Words	Accuracy
SIFT Bag of Words	50	21.67%
SIFT Bag of Words	100	23.13%
SIFT Bag of Words	150	21.47%
SIFT Bag of Words	200	20.40%
ORB Bag of Words	50	24.07%
ORB Bag of Words	100	22.73%
ORB Bag of Words	150	22.40%
ORB Bag of Words	200	21.87%

Model	Num_Clusters	Accuracy
SIFT Fisher Vector	25	14.07%
SIFT Fisher Vector	50	18%
SIFT Fisher Vector	100	15.73%
ORB Fisher Vector	25	20.60%
ORB Fisher Vector	50	24.09%
ORB Fisher Vector	100	22.67%

Key Takeaways:

- SIFT was far more computationally expensive
- With BoW SIFT was more suited
- After 125 Key words overfitting happened
- The statistical power of GMM proved useful
- ORB and fisher vector outperformed other
- Still poor results with depedency on certain features and prone to default to guessing same label.

Linear Neural Network and Convolutional Neural Network Training Phase

Preprocessing:

- Dataset had mix of greyscale and RGB
- Converted all to RGB
- Batched into default size of 32 using DataLoader
- Transformed to tensor and normalised data

```
param_grid = {
    'learning_rate': [0.001, 0.01],
    'batch_size': [32,64,128,256],
    'dropout_rate': [0.3,0.5],
    'num_epochs': [40,50,60],
    'optimizer': ['adam', 'sgd']  #
}
```

Training Process:

1. Start with a base model with simple number of layers.
2. Find downfalls and try improve on previous
3. Create a set of models and test
4. Choose a model to take for further improvement
5. Apply techniques like early stopping, hyperparameter tuning and batch normalisation

```
Early stopping triggered after epoch 15
Training with params: {'batch_size': 32, 'dropout_rate': 0.3, 'learning_rate': 0.001, 'num_epochs': 40, 'optimizer': 'sgd'}
Epoch 1/40 - Train Accuracy: 14.56% | Train Loss: 2.5931 | Val Accuracy: 23.47% | Val Loss: 2.4200
Epoch 2/40 - Train Accuracy: 23.33% | Train Loss: 2.4049 | Val Accuracy: 25.13% | Val Loss: 2.3331
Epoch 3/40 - Train Accuracy: 26.47% | Train Loss: 2.3068 | Val Accuracy: 28.27% | Val Loss: 2.2701
Epoch 4/40 - Train Accuracy: 29.58% | Train Loss: 2.2259 | Val Accuracy: 30.20% | Val Loss: 2.2298
Epoch 5/40 - Train Accuracy: 31.40% | Train Loss: 2.1584 | Val Accuracy: 30.40% | Val Loss: 2.1934
Epoch 6/40 - Train Accuracy: 33.47% | Train Loss: 2.0982 | Val Accuracy: 30.87% | Val Loss: 2.1891
Epoch 7/40 - Train Accuracy: 35.60% | Train Loss: 2.0374 | Val Accuracy: 31.53% | Val Loss: 2.1789
Epoch 8/40 - Train Accuracy: 38.02% | Train Loss: 1.9668 | Val Accuracy: 30.67% | Val Loss: 2.1668
Epoch 9/40 - Train Accuracy: 38.40% | Train Loss: 1.9321 | Val Accuracy: 32.33% | Val Loss: 2.1523
Epoch 10/40 - Train Accuracy: 40.84% | Train Loss: 1.8617 | Val Accuracy: 32.47% | Val Loss: 2.1545
Epoch 11/40 - Train Accuracy: 42.84% | Train Loss: 1.8174 | Val Accuracy: 33.00% | Val Loss: 2.1569
Epoch 12/40 - Train Accuracy: 45.71% | Train Loss: 1.7549 | Val Accuracy: 32.87% | Val Loss: 2.1523
Epoch 13/40 - Train Accuracy: 45.18% | Train Loss: 1.7179 | Val Accuracy: 33.00% | Val Loss: 2.1751
Early stopping triggered after epoch 14
Training with params: {'batch_size': 32, 'dropout_rate': 0.3, 'learning_rate': 0.001, 'num_epochs': 50, 'optimizer': 'adam'}
Epoch 1/50 - Train Accuracy: 11.76% | Train Loss: 3.1682 | Val Accuracy: 15.13% | Val Loss: 2.6058
Epoch 2/50 - Train Accuracy: 15.71% | Train Loss: 2.6329 | Val Accuracy: 19.47% | Val Loss: 2.5227
Epoch 3/50 - Train Accuracy: 17.60% | Train Loss: 2.5758 | Val Accuracy: 19.47% | Val Loss: 2.5353
Epoch 4/50 - Train Accuracy: 19.27% | Train Loss: 2.5466 | Val Accuracy: 20.93% | Val Loss: 2.4841
Epoch 5/50 - Train Accuracy: 20.56% | Train Loss: 2.5049 | Val Accuracy: 20.47% | Val Loss: 2.4777
Epoch 6/50 - Train Accuracy: 21.36% | Train Loss: 2.4741 | Val Accuracy: 22.27% | Val Loss: 2.4530
Epoch 7/50 - Train Accuracy: 20.73% | Train Loss: 2.4629 | Val Accuracy: 23.07% | Val Loss: 2.4311
Epoch 8/50 - Train Accuracy: 21.89% | Train Loss: 2.4566 | Val Accuracy: 21.67% | Val Loss: 2.4551
Epoch 9/50 - Train Accuracy: 22.29% | Train Loss: 2.4329 | Val Accuracy: 20.47% | Val Loss: 2.4413
Epoch 10/50 - Train Accuracy: 21.89% | Train Loss: 2.4435 | Val Accuracy: 22.27% | Val Loss: 2.4055
Epoch 11/50 - Train Accuracy: 22.02% | Train Loss: 2.4201 | Val Accuracy: 22.67% | Val Loss: 2.4278
Epoch 12/50 - Train Accuracy: 22.27% | Train Loss: 2.4202 | Val Accuracy: 20.73% | Val Loss: 2.4654
Epoch 13/50 - Train Accuracy: 22.36% | Train Loss: 2.4121 | Val Accuracy: 22.13% | Val Loss: 2.4325
Epoch 14/50 - Train Accuracy: 22.62% | Train Loss: 2.4100 | Val Accuracy: 20.93% | Val Loss: 2.4315
Epoch 15/50 - Train Accuracy: 24.40% | Train Loss: 2.3619 | Val Accuracy: 22.13% | Val Loss: 2.4030
Epoch 16/50 - Train Accuracy: 23.33% | Train Loss: 2.3577 | Val Accuracy: 20.40% | Val Loss: 2.4700
Epoch 17/50 - Train Accuracy: 25.11% | Train Loss: 2.3494 | Val Accuracy: 21.60% | Val Loss: 2.4067
Epoch 18/50 - Train Accuracy: 23.82% | Train Loss: 2.3690 | Val Accuracy: 17.67% | Val Loss: 2.4846
Epoch 19/50 - Train Accuracy: 24.78% | Train Loss: 2.3368 | Val Accuracy: 18.33% | Val Loss: 2.4929
Early stopping triggered after epoch 20
Training with params: {'batch_size': 32, 'dropout_rate': 0.3, 'learning_rate': 0.001, 'num_epochs': 50, 'optimizer': 'sgd'}
Epoch 1/50 - Train Accuracy: 14.60% | Train Loss: 2.6030 | Val Accuracy: 23.93% | Val Loss: 2.4317
Epoch 2/50 - Train Accuracy: 22.58% | Train Loss: 2.4098 | Val Accuracy: 27.67% | Val Loss: 2.3272
Epoch 3/50 - Train Accuracy: 25.73% | Train Loss: 2.3208 | Val Accuracy: 29.93% | Val Loss: 2.2639
Epoch 4/50 - Train Accuracy: 29.71% | Train Loss: 2.2232 | Val Accuracy: 30.40% | Val Loss: 2.2212
Epoch 5/50 - Train Accuracy: 31.62% | Train Loss: 2.1615 | Val Accuracy: 30.67% | Val Loss: 2.2023
```

Techniques Used:

1. Early Stopping with Patience
2. Hyperparameter Tuning:
   - Optimiser Type
   - Learning Rate
   - Batch Size
   - Number of epochs
   - Dropout Rate
   - Learning Rate Scheduler
3. Data Augmentation
4. Batch Normalisation
5. Global Average Pooling (CNN)

Linear Neural Network

Models:

LinearNNV1: Baseline

- Input Size: 12288 (Flattened Image)
- Hidden Layers: 2 → [512, 256]
- Activation: ReLU
- Dropout: 0.5 (after each layer)
- Output: 15 neurons (for 15 classes)

LinearNNV2: Adding Layers

- Hidden Layers: 4 → [1024, 512, 256, 128]
- Dropout: 0.3

LinearNNV3: Improved Feature Extraction

- Hidden Layers: 6 → [2048, 1024, 512, 256, 128, 64]
- Dropout Strategy: Gradually decreasing (0.5→ 0.2)
- Goal: Better generalisation with strong feature learning

Final Model

- V2
- Early Stopping
- SGD (0.001 LR)
- Batch Size =128

Model (architecture)	Epochs	Batch_Size	Optimiser	Data Augmentation	Early Stopping	Hyperparameter Tuning	Training Acc	Validation Acc	Test Acc	Test Loss
V1	40	32	Adam(lr=0.001)	No	No	No	27.73%	19.20%	22.60%	2.4754
V2	40	32	Adam(lr=0.001)	No	No	No	30.18%	18.07%	18.20%	2.6662
V3	40	32	Adam(lr=0.001)	No	No	No	33.93%	23.93%	25.73%	2.6344
V1	50	64	SGD(lr=0.001 Dropout=0.3)	No	Yes	Yes	48.31%	34.26%	33.47%	2.1243
V2	60	128	SGD(lr=0.01 Dropout=0.3)	No	Yes	Yes	39.73%	32.47%	31.53%	2.1900
V3	40	32	SGD(lr=0.01) Dropout=0.3	No	Yes	Yes	38.76%	28.13%	26.54%	2.4587

Model (architecture)	Epochs	Batch_Size	Optimiser	Data Augmentation	Early Stopping	Hyperparameter Tuning	Training Acc	Validation Acc	Test Acc	Test Loss
V2	60	128	SGD(0.001, momentum=0.9) Dropout Rate=0.3	No	Yes (Patience=5)	Yes	45.42%	32.47%	33.00%	2.1522
V2	60	128	SGD(0.001, momentum=0.9) Dropout Rate=0.5	No	Yes (Patience=5)	Yes	32.09%	29.73%	30.00%	2.1893
V2	100	128	SGD(0.001, momentum=0.9) Dropout Rate=0.3	Yes	Yes (Patience=5)	Yes	44.20%	28.93%	30.47%	2.2116
V2	100	128	SGD(0.001, momentum=0.9) Dropout Rate=0.5	Yes	Yes (Patience=5)	Yes	35.00%	27.33%	27.67%	2.2888

Convolutional Neural Network

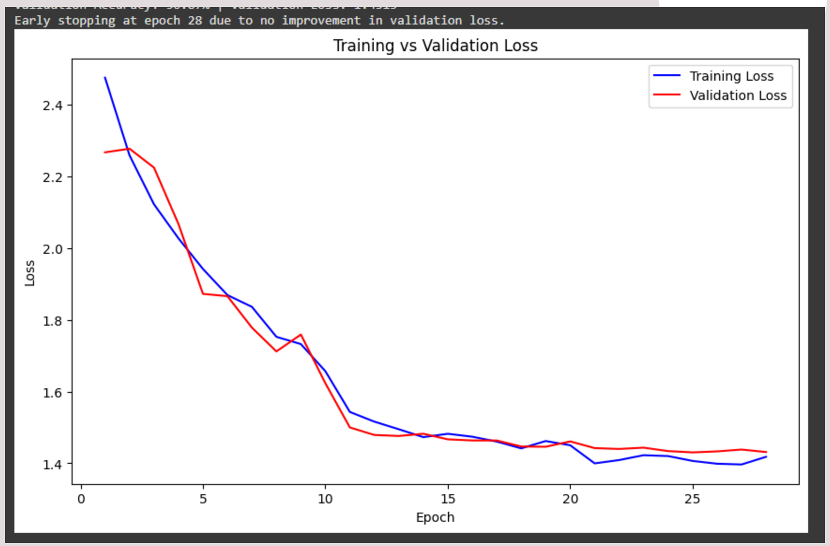
Had a varying number of models ranging from V1-V8

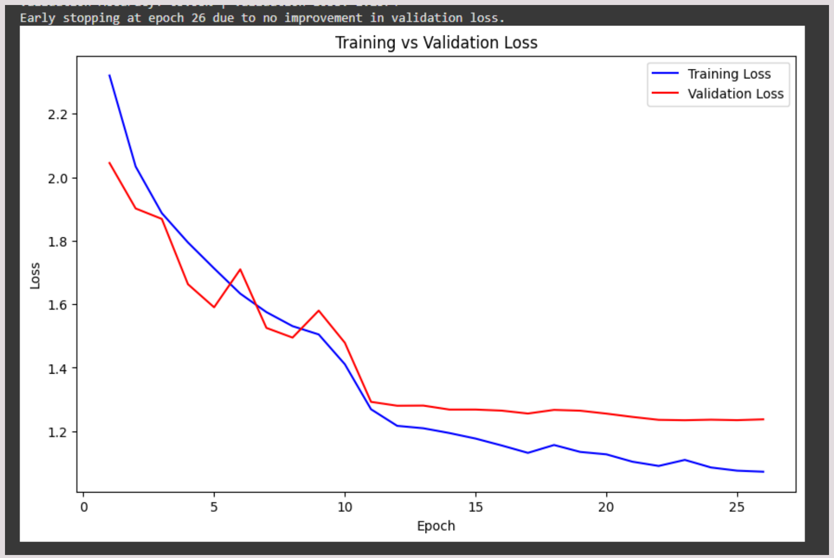
V1-V5 focused on varying number of convolutional layers, dropout rates and fully connected layers

V6-V8 built on this but incorporated batch normalisation and global average pooling

Key Takeaways

- Early Stopping was effective
- Four Convolutional Layers was best amount
- Battle between overfitting and performance
- Adam Optimiser best with small learning rate
- Augmentation improved fit





Comparison of Best Models

Model	Epochs/Number of Clusters	Architecture	Hyperparameters	Test Accuracy
ORB Fisher Vector	50	Fitted Gaussian Mixture Model Classified using LinearSVC() Output (15 classes)	Max_iter=1000, C=0.1	24.09%
LinearNNV2	60	Input Size - 64 x 64 x 3 Flattening Layer - (12288) Four Hidden Layers = [1024, 512,256,128] ReLU Activation Dropout - 30% Output(15 classes)	Batch_size = 128 Optimiser = SGD Learning Rate=0.001 Momentum = 0.9 Loss = CrossEntropy Dropout Rate=0.5 Early Stopping Patience = 5	33.00%
CNNV7	100	Input (64×64×3) → Conv1 (16 filters, 3×3, ReLU, BN, MaxPool 2×2) → Conv2 (32 filters, 3×3, ReLU, BN, MaxPool 2×2) → Conv3 (64 filters, 3×3, ReLU, BN, MaxPool 2×2) → Conv4 (128 filters, 3×3, ReLU, BN, MaxPool 2×2) → Global Avg Pool → Fully Connected Layer (128 → 15) → Dropout (20%) → Output (15 classes). BN (Batch Normalisation)	Batch_size= 64 Optimiser = Adam Learning Rate=0.001 Scheduler = StepLR(10,0.1) Loss=CrossEntropy Dropout Rate = 0.2 Early Stopping Patience=3 Data Augmentation=No	62.27%
CNNV7 Data Augmentation	100	Same as above	Same as above except Data Augmentation=True	58.67% (Neater Train/Val loss plot)

Error Analysis

Linear NN vs Handcrafted Features

•Handcrafted Model: Relied on dominant features → High recall, low precision (e.g., guinea pigs & spiny lobsters).
•Linear NN: Learned more balanced features → Better precision & recall, improved performance on labels like 2 & 13.

Linear NN vs CNNs

•Linear NN Mistakes: Predicted chimpanzee (4) as backpack (14) → Features not informative.
•CNN Advantage: Avoided major mistakes, instead misclassifying boa constrictor (1) as spiny lobster (7) → More logical errors based on shared visual features.

Key Test Case: Bucket & Pail (Label 3)

•Worst-performing category across all models due to distractions in images.
•Handcrafted Model: Focused on irrelevant features (e.g., eyes).
•CNN: Extracted useful spatial features, leading to far more correct predictions.