EEEM066 Knife classification

Keith Arogo Owino, URN: 6835251, ko00537@surrey.ac.uk

Abstract

This report delves into a comprehensive exploration of different deep learning models, specifically convolutional neural networks. Both transfer learning methods and models that learn from 'scratch' are explored. The primary objective is to identify potentially suitable models and enhance each model's performance by fine-tuning key hyperparameters and altering certain architectural components. The best version of each model is also compared against all the others to establish a hierarchy of suitability for the task under very specific conditions. The study employs a systematic approach to evaluate the impact of different values of these hyperparameters and different architectural changes on a model's overall accuracy, using mean Average Precision (mAP), top 1 accuracy and top 5 accuracy as the primary metrics. The report encompasses a detailed analysis of various architectures, including a custom non-transfer learning architecture, EfficientNet, EfficientNetV2, and ResNet, each offering unique results considering their varying architectures. The report also explores the potential of ensemble methods, aiming to combine strengths of the best two models to achieve even better results. This report is able to draw vivid conclusions with regard to different transfer learning methods and their suitability for fine-grained knife-classification. This report also produces insights into the differences between transfer learning methods and non-transfer learning methods in terms of performance.

1. Introduction

CNNs are a critical aspect in the field of deep learning, especially in image processing tasks. The choice of model and the fine-tuning of certain hyperparameters can significantly influence the learning process and the overall effectiveness of a CNN model. This report presents an extensive analysis of the suitability of various models for knife classification including a custom non-transfer learning model, the EfficientNet series, its evolution EfficientNetV2, the ResNet models and ensemble models. It also comprehensively looks at the effects of varying various hyperparameters and architectural components on their performance.

During the hyperparameter optimization process, the experiment begins by establishing a baseline using default hyperparameter values, followed by an exploration of different and values. Early stopping and learning rate adjustment techniques, such as the Cosine Annealing Learning Rate Scheduler, are implemented to optimize the training process.

The report also investigates the synergy of ensemble architectures that combine features of different models, aiming to leverage their collective strengths [1].

In the concluding sections, the report synthesizes findings from the experiments, comparing the performance of different models and hyperparameter configurations. The goal is to provide comprehensive insights and practical recommendations aiding in the selection of optimal hyperparameters for knife classification challenges.

2. Custom CNN.

This section is concerned with the implementation and experimentation of the model that performs training from 'scratch'. There are 6 hyper-parameters. Only the batch size, the learning rate and the number of epochs were considered tunable hyperparameters throughout the experiments. These were the default values for all the hyperparameters which formed the baseline for the hyperparameter tuning. The Number of classes is 192, Image weight is 224, Image height is 224, Batch size of 16, 20 epochs and a Learning rate of 0.00005.

2.1. Structure.

This section is concerned with the inner structure and configuration of the model.

2.1.1 Layer Configuration.

Comprises of three convolutional layers each designed to process knife image data through filter kernels.
First layer with 3 input channels (suitable for RGB images) and 32 output channels, using kernels of size 3x3, a stride of 1, and padding of 1.
Second layer taking 32 input channels, outputting 64 channels with similar kernel size, stride, and padding.
Third layer with 64 input channels, escalating to 128 output channels.
Batch Normalization Layers are also applied after each convolutional layer to normalize the outputs, stabilizing and expediting the training process.

2.1.2 Pooling Layer.

Max pooling was used, employing a 2x2 window for down sampling, reducing each layer's spatial dimensions to lower computational requirements and mitigate overfitting.

2.1.3 Fully connected layer

The 1st layer transforms the flattened feature maps into a 512-dimensional vector with the input from the final pooling layer to be of dimensionality 128 * 28 * 28.
The 2nd layer maps the 512-dimensional vector to the final output size, equal to the number of classes. The ReLu activation function was used.

2.2. Hyperparameter tuning.

2.2.1 Learning Rate

The learning rate is a hyperparameter that determines the size of the steps taken during the optimization process. It controls how much the model's weights are updated during training[2].

2.2.2 Cosine Annealing Learning Rate Scheduler.

The Cosine Annealing Scheduler adjusts the learning rate following a cosine curve. It starts with a higher learning rate and gradually reduces it, following a cosine function, until it reaches a minimum value[3].
Avoiding Local Minima: The cosine annealing method helps in avoiding local minima in the loss landscape, as the learning rate 'cycles' can help the model to 'jump out' of local minima. As the learning rate decreases, the steps in the weight space become smaller, allowing for more refined exploration near the end of training[3].

2.2.2.1 Learning rate experiments

Execution	**Learning Rate**	**Batch Size**	**Epochs**	**Best mAP**
1	0.00005	16	10	0.1006
2	**0.0001**	**16**	**19**	**0.1321**
3	0.0005	16	6	0.0578
4	0.001	16	5	0.0060

Table 1 - Learning rate experiments - custom CNN

Based on the results a learning rate of 0.0001 was utilized.

2.2.3 Batch Size

Batch size refers to the number of training examples utilized in one iteration of model training. In each epoch, the dataset is divided into batches, and the model's weights are updated after processing each batch[4].

Large Batch Size: Can lead to faster training in terms of computation time[4].

Small Batch Size: Tends to have noisier gradient estimates, which can help escape local minima. Often leads to better generalization but can be computationally less efficient.[4].

2.2.3.1 Batch size experiments

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.0001	4	6	0.0595
2	**0.0001**	**8**	**20**	**0.1576**
3	0.0001	16	19	0.1321
4	0.0001	32	15	0.1001

Table 2 - Batch size experiments - custom CNN

Based on the results, a batch size of 8 was utilized.

2.2.4 Top 1 and Top 5 Accuracy – Custom CNN

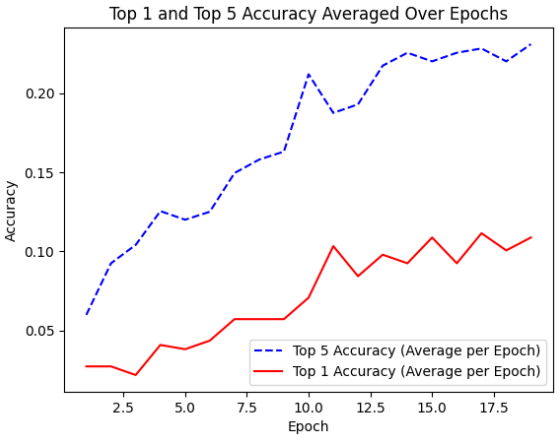

Figure 1 - Top 1 and Top 5 accuracy - custom CNN

Based on Figure 1 the model is learning over time, as evidenced by the increase in both Top 1 and Top 5 accuracies.

Top 5 accuracy is consistently higher than Top 1 accuracy, which is expected since it's easier for the correct prediction to fall within the top five guesses than to be the single top guess.

There is a noticeable gap between Top 1 and Top 5 accuracies, suggesting that while the correct prediction may not always be the model's first choice, it often ranks it highly.

3. Transfer Learning Models.

This section is concerned with the EfficientNet series, its evolution EfficientNetV2, the ResNet models and an ensemble model.

3.1. Baseline model and EffecientNet-B0 experiments.

This section demonstrates implementations and experiments that were performed based on the baseline model's performance.

3.1.1 Early Stopping

Figure 2 below shows a plot of the validation mAP per epoch obtained after training the baseline model with the default parameters. The mAP starts from 0.304 and shows consistent improvement up to the 10th epoch, reaching a peak of 0.667.

After the 10th epoch, there is a fluctuation in the mAP values. The mAP peaks at 0.667 at the 10th epoch, then drops and occasionally rises slightly, but doesn't exceed this peak.

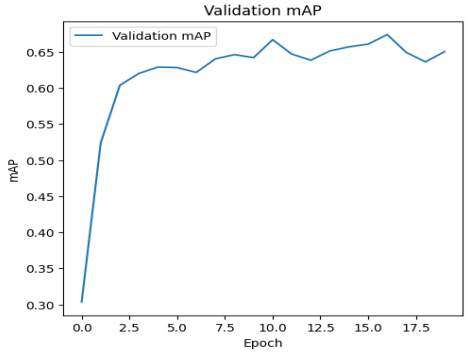

Figure 2 - validation map per epoch for baseline model

3.1.2 Patience Parameter Value

The patience parameter determines how many epochs you allow without improvement before halting training. Given the highest peak is at epoch 10 and there's no significant improvement after this, a patience of 5 epochs was chosen.

3.1.3 Min Delta Value

The min delta is the smallest change in the mAP considered as an improvement. From figure 1, changes after epoch 10 are minor. A min delta of 0.02 was considered appropriate.

3.1.4 Results with different learning rates.

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.00005	16	20	0.6668
2	**0.0001**	**16**	**17**	**0.7090**
3	0.0005	16	7	0.6408
4	0.001	16	8	0.6002
5	0.005	16	16	0.3702
6	0.01	16	13	0.3317

Table 3 - learning rate experiments for the baseline model

Execution 1 with a learning rate of 0.00005 over 20 epochs showed a decent mAP of 0.6668, suggesting that a lower learning rate can be beneficial over more epochs.

Execution 2's increased learning rate to 0.0001 for 17 epochs achieved the highest mAP of 0.7090, indicating that a moderate learning rate can enhance performance within fewer epochs. However, further increases in learning rate to 0.0005 and 0.001 resulted in lower mAPs of 0.6408 and 0.6002, respectively, suggesting too rapid convergence and missed details. Executions 5 and 6 with even higher rates of 0.005 and 0.01 confirmed this trend.

3.1.5 Results with different batch sizes.

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.0001	2	7	0.6435
2	0.0001	4	7	0.7053
3	**0.0001**	**8**	**10**	**0.7314**
4	0.0001	16	17	0.7090
5	0.0001	32	10	0.6610
6	0.0001	64	11	0.6315

Table 4 - Batch size experiments for the baseline model

The highest mAP achieved was 0.7314 in Execution 3, using a batch size of 8 and 10 epochs.

As the batch size increases, there seems to be a trend where the mAP first increases up to the 3rd execution and from this point it then decreases. Execution 3 suggests a balance between accurate gradient estimation, generalization, and effective utilization of computational resources.

3.2. EfficientNet B4, EfficientNetV2 models, ResNet models and the ensemble model hyperparameter experiments.

This section is concerned with the experiments and results obtained for different hyperparameter values for the EfficientNet series, its evolution EfficientNetV2, the ResNet models and the ensemble model.

3.2.1 Experiments results and discussion.

The baseline model used weights from EffecientNet-B0. All EffecientNet model iterations were tested from B0 to B7 and the most suitable for this task was concluded to be EffecientNet-B4.

The optimal hyperparameters of the previous model in the testing pipeline were used as the baseline for testing the current model and then applying specific adjustments.

EffecientNet B4

Learning Rate

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.00005	8	8	0.7509
2	**0.0001**	**8**	**8**	**0.7800**
3	0.0005	8	8	0.6512
4	0.001	8	8	0.5740

Batch Size Tuning

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.0001	2	7	0.7444
2	0.0001	4	7	0.7755
3	**0.0001**	**8**	**8**	**0.7800**
4	0.0001	16	8	0.7385
5	0.0001	32	12	0.7178
6	0.0001	64	12	0.7107

EffecientNetV2-L

Learning Rate

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.00005	8	8	0.7639
2	**0.0001**	**8**	**6**	**0.7909**
3	0.0005	8	14	0.7169
4	0.001	8	12	0.6170

Batch Size

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.0001	2	7	0.7555
2	0.0001	4	8	0.7805
3	**0.0001**	**8**	**6**	**0.7909**
4	0.0001	16	8	0.7585
5	0.0001	32	12	0.7309
6	0.0001	64	12	0.7230

ResNet-152

Learning Rate

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.00001	8	12	0.6512
2	**0.00005**	**8**	**9**	**0.8009**
3	0.0001	8	8	0.7702
4	0.0005	8	8	0.6253
5	0.001	8	7	0.6000

Batch Size

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	0.00005	2	9	0.7540
2	0.00005	4	9	0.7772
3	**0.00005**	**8**	**9**	**0.8009**
4	0.00005	16	13	0.7358
5	0.00005	32	13	0.7185
6	0.00005	64	14	0.7019

Ensemble Model

Learning Rate

Execution	Learning Rate	Batch Size	Epochs	Best mAP
1	**0.00005**	**8**	**9**	**0.8349**
2	0.0001	8	8	0.7880

Table 5 - Experimental results for all models

Table 5 provides an overview of hyperparameter tuning results for different neural network architectures, namely EfficientNet B4, EfficientNetV2-L, ResNet-152, and an Ensemble Model, focusing on learning rate and batch size adjustments.

For EfficientNet B4, a higher learning rate (0.0001) combined with a moderate batch size (8) yielded the best mAP (0.7800), indicating a balance between learning rate and batch size is crucial for optimal performance. However, as the learning rate increased beyond this point, the mAP significantly decreased, suggesting that too high a learning rate might lead to poorer model convergence. In batch size tuning, a similar pattern emerges with a batch size of 8 and a learning rate of 0.0001 providing the highest mAP (0.7800), showing that larger batch sizes do not necessarily improve performance and can lead to a decline in model accuracy.

EfficientNetV2-L shows a similar trend, with its optimal performance (0.7909 mAP) at a learning rate of 0.0001 and batch size of 8.

The ResNet-152 model, interestingly, achieved its best mAP (0.8009) at a lower learning rate (0.00005) and a moderate batch size (8), suggesting different optimal settings for different architectures.

The Ensemble Model demonstrated the highest mAP of all the models at a learning rate of 0.00005 and a batch size of 8, underlining the effectiveness of ensemble approaches in improving model performance.

3.3. EfficientNet Model.

This section is concerned with the analysis of the EffecientNet models. It demonstrates how the best EffecientNet model was distinguished from the others. It then goes further and analyses the best EffecientNet model (EffecientNet-B4) using an epoch wise plot of the top 1 and top 5 accuracy.

3.3.1 Results with different EffecientNet models.

Model	Best mAP
EfficientNet-B0	0.7314
EfficientNet-B1	0.7198
EfficientNet-B2	0.7359
EfficientNet-B3	0.7395
EfficientNet-B4	**0.7757**
EfficientNet-B5	0.7353
EfficientNet-B6	0.7714
EfficientNet-B7	0.7182

Table 6 - Experiments with different EffecientNet models.

3.3.2 Top 1 and Top 5 Accuracy - EffecientNet-B4.

Top 1 Accuracy: In figure 3 the solid red line shows the Top 1 accuracy. After epoch 4, there's a noticeable dip in epoch 5, followed by a recovery in epoch 6, and finally, it drops again in epochs 7 and 8.

Top 5 Accuracy: The blue dashed line represents the Top 5 accuracy. The graph shows that the Top 5 accuracy starts at around 82%, increases to a peak of approximately 87% by epoch 2, and then fluctuates slightly but stays relatively stable across the remaining epochs.

The Top 1 accuracy is more variable over epochs compared to the Top 5 accuracy. This indicates that while the model is consistent at ranking the correct label within its top 5 predictions, its confidence in the most likely prediction is less stable.

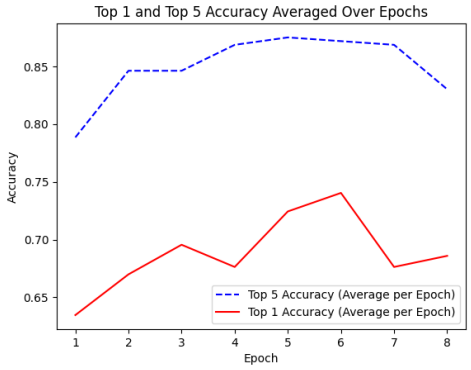

Figure 3 - Top 1 and Top 5 Accuracy for EffecientNet-B4's best configuration

3.4. EffecientNetV2-L

This section is concerned with the analysis of the EffecientNetV2 models. It demonstrates how the best EffecientNetV2 model was distinguished from the others. It then goes further and analyses the best EffecientNetV2 model (EffecientNetV2-L) using an epoch wise plot of the top 1 and top 5 accuracy.

3.4.1 Results with different EfficientNetV2 Models.

Model	Best mAP
EfficientNetV2-S	0.7356
EfficientNetV2-M	0.7815
EfficientNetV2-L	**0.7998**

Table 7 - Experiments with different EffecientNetV2 models.

3.4.2 Top 1 and Top 5 Accuracy - EfficientNetV2-L

As shown in figure 4 the Top 1 accuracy starts just above 70% in the first epoch and shows a declining trend over the 6 epochs, ending just above 65% in the last epoch. This decline suggests that the model's ability to correctly predict the exact class on the first try is decreasing as training progresses.

The Top 5 accuracy begins at around 85% and experiences a slight increase, peaking near 87%. After the peak, it has a minor dip in epoch 3, then stabilizes and has a slight uptick by epoch 6. This indicates that while the model's top prediction is becoming less accurate, it still often includes the correct prediction within its top five predictions.

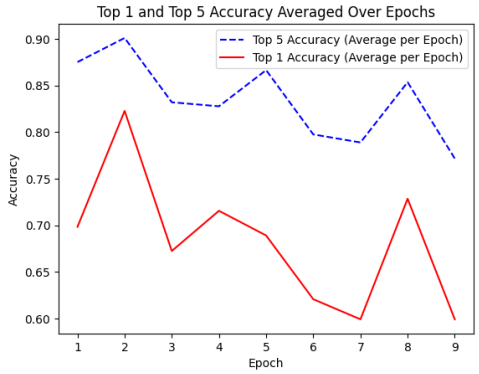

Figure 4 - Top 1 and Top 5 Accuracy for EffecientNetV2-L's best configuration

3.5. ResNet

This section is concerned with the analysis of the ResNet models. It demonstrates how the best ResNet model was distinguished from the others. It then goes further and analyses the best ResNet model (ResNet - 152) using an epoch wise plot of the top 1 and top 5 accuracy.

3.5.1 Results with different ResNet models

Model	Best mAP
ResNet-18	0.6000
ResNet-34	0.5858
ResNet-50	0.7338
ResNet-101	0.7107
ResNet-152	**0.7702**

Table 8 - Results with different ResNet models

In comparing different ResNet architectures for knife classification, ResNet-18's fewer layers may not have detected complex features adequately. ResNet-34's slight complexity increase didn't yield better performance, suggesting the change wasn't significant enough change in the number of layers. ResNet-50, however, with its greater depth and bottleneck layers, showed substantial mAP improvements, indicating its ability to learn more complex patterns efficiently. ResNet-152 achieved the highest mAP, benefiting from its depth in learning very complex data features more effectively than the shallower models [5].

3.5.2 Top 1 and Top 5 Accuracy – ResNet - 152.

As seen in Figure 5 the top 1 Accuracy starts below 40% in the first epoch, showing a rapid improvement by epoch 2. The accuracy then levels off and maintains a relatively stable performance, with minor fluctuations around 70% from epochs 3 to 9.

The Top 5 accuracy starts at around 60% in the first epoch and experiences a sharp increase, reaching approximately 90% by epoch 3. From epoch 3 onwards, it remains consistent, with a slight decline in epochs 8 and 9 but still stays above 85%.

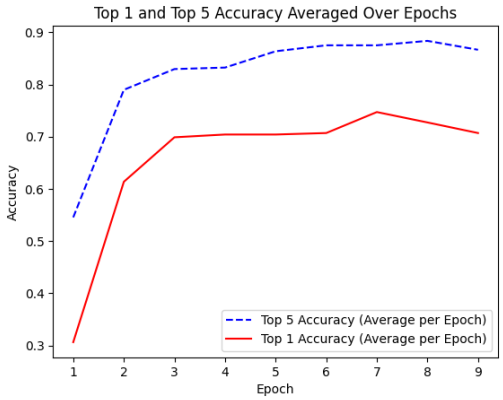

Figure 5 - Top 1 and Top 5 Accuracy for the best resNet-152 model

3.6. Ensemble Architecture

The ensemble model concatenates the features from both EfficientNetV2-L and ResNet-152 as they proved to be the two best models throughout the experiments. The final classifier layer combines these features for classification.

3.6.1 Top 1 and Top 5 Accuracy

As seen in figure 5 starting from epoch 1, the Top 1 accuracy is roughly 75% and shows minor fluctuations throughout the training process, ending just below 80% by epoch 9. The trend is mostly stable with a slight increase over time, suggesting that the model is learning and slightly improving its ability to correctly identify the most probable class. The Top 5 accuracy starts just below 90% and rises to about 90% by epoch 2, then exhibits a gradual decline, finishing around 87.5%. The initial high accuracy indicates that the model is quite good at including the correct class in its top five predictions.

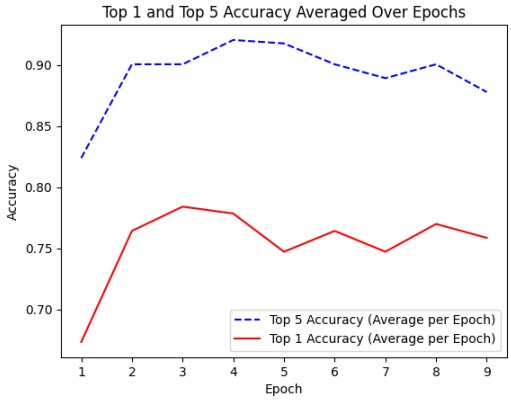

Figure 6 - Top 1 and Top 5 Accuracy for the best ensemble model

4. Comparative Analysis and Conclusion

Model	Epochs	Batch Size	Learning Rate	Highest mAP
Ensemble	**9**	**8**	**0.00005**	**0.8349**
EffecientNetV2-L	6	8	0.0001	0.7909
ResNet-152	9	8	0.00005	0.8009
EfficientNet-B4	8	8	0.0001	0.7800
Custom CNN	20	8	0.0001	0.1576

Table 9 - Comparing the best models.

The ensemble model, leveraging the combined capabilities of EfficientNetV2-L and ResNet-152, demonstrates the highest performance for the knife classification task. This suggests that in cases where individual models show competitive but distinct strengths, an ensemble approach can be a powerful strategy to maximize performance.

It's noteworthy that the custom model, the only one in our study not utilizing transfer learning, showed significantly lower results. This demonstrates the impact of transfer learning in enhancing model performance especially in situations where time and computational resources might be limited.

5. References

[1] S. Shamshirband, E. Jafari Nodoushan, J. E. Adolf, A. Abdul Manaf, A. Mosavi, and K. wing Chau, "Ensemble models with uncertainty analysis for multi-day ahead forecasting of chlorophyll a concentration in coastal waters," *Engineering Applications of Computational Fluid Mechanics*, vol. 13, no. 1, 2019, doi: 10.1080/19942060.2018.1553742.

[2] J. Jepkoech, D. M. Mugo, B. K. Kenduiywo, and E. C. Too, "The Effect of Adaptive Learning Rate on the Accuracy of Neural Networks," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 8, 2021, doi: 10.14569/IJACSA.2021.0120885.

[3] H. Xie and Z. Wu, "A robust fabric defect detection method based on improved refinedet," *Sensors (Switzerland)*, vol. 20, no. 15, 2020, doi: 10.3390/s20154260.

[4] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, 2020, doi: 10.1016/j.icte.2020.04.010.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016. doi: 10.1007/978-3-319-46493-0_38.