Module	Fundamentals of Machine Learning: EEEM066
Assessment type	Coursework assignment
Weighting	20%
Deadline	4:00PM, Friday, 6 December 2024 (Week 11)
STUDENT URN	
STUDENT Name	Dhruvraj Singh Rawat

Objectives:

- Learn and become familiar with the process of sequential hyperparameter tuning of an entire deep learning pipeline.
- Develop intuition via experimentation on how to use such model design and parameter tuning to improve the performance of a pipeline.
- Interpret experimental results of deep learning experiments and their implications.

Outcomes:

- Navigate general deep learning codebases and modify according to requirements.
- Empirically suggest reasonable values for hyperparameters and select the ones enabling the best performance.
- Understand the implications of changing the backbone architecture of a pipeline.
- Reason why data augmentation works.
- Understand the implications and importance of selecting the best learning rate and batch size.
- Understand how to report experimental results professionally. This includes using tables, plots, and diagrams to communicate ideas and results.

Knife classification in real-world images

For this coursework assignment, you are encouraged to apply what you have learnt from the module materials and your participation in collaborative learning activities like class discussions.

Guidelines

- Submit your assignment as a **Word or PDF document**, along with the supporting evidence as required, via the SurreyLearn submission page before the deadline. Do not submit any other unrelated documents that could flood your submission and mislead the marking process. Only the latest version submitted before the deadline will be assessed if multiple versions exist.
- Label your submission file as: EEEM066-[insert your URN number]-Assignment e.g., EEEM066-1234567-Assignment
- Include a reference list at the end of your assignment, and use the Harvard or IEEE referencing style when citing sources. Please use the selected option consistently.
- **Word limit:** Each section of the report should not exceed **200 words** (excluding tables, plots, and graphs). Penalties of up to 50% may be applied for unnecessarily long reports.
- This assignment requires time and effort. It is recommended that you start working on it early to minimise the possibility of experiencing stress around it and avoid missing the deadline.
- You will receive both an overall mark and written feedback on your assignment.
- Extensions for this assignment will only be granted under exceptional circumstances. Extension requests must be submitted via the Extenuating Circumstances (ECs) process, as tutors are not permitted to approve ad hoc extension requests.

Note:
Please follow Surrey's guidelines for avoiding plagiarism.
You must complete this coursework individually. Copying code or text from the internet or another student and including it in your assignment without proper attribution constitutes plagiarism.
Undetected plagiarism undermines the quality of your degree by hindering the assessor's ability to evaluate your work fairly and prevents you from learning through the coursework. If plagiarism is suspected, you will be referred to an Academic Misconduct Panel, which may result in academic sanctions.
All assignments will be checked for plagiarism using Turnitin. Therefore, it is important that you correctly reference sources that have informed your work where appropriate.

Task

Review the instructions for the assignment carefully before starting. This assignment is marked out of 100 and will contribute **20%** to your total module grade. The three sections of the assignment are worth **90 marks**, with an additional **10 marks** allocated to the presentation and clarity of the report.

Background

In recent years, the number of crimes involving sharp objects, such as knives, has increased all across the world. In the year ending March 2022, England and Wales saw around 45,000 offences involving a knife or sharp instrument. This was 9% higher than in 2020/21 and 34% higher than in 2010/11.

Therefore, there is a need to develop an AI-based weapon analysis system that enables officers to identify the make, model, and seller of knives using photographs. Although functional, current knife recognition and labelling methods are beset by inefficiencies due to strained public resources, the possibility of human error, and the UK's expanding digital footprint.

The following figure provides a visual representation of a knife classification system.

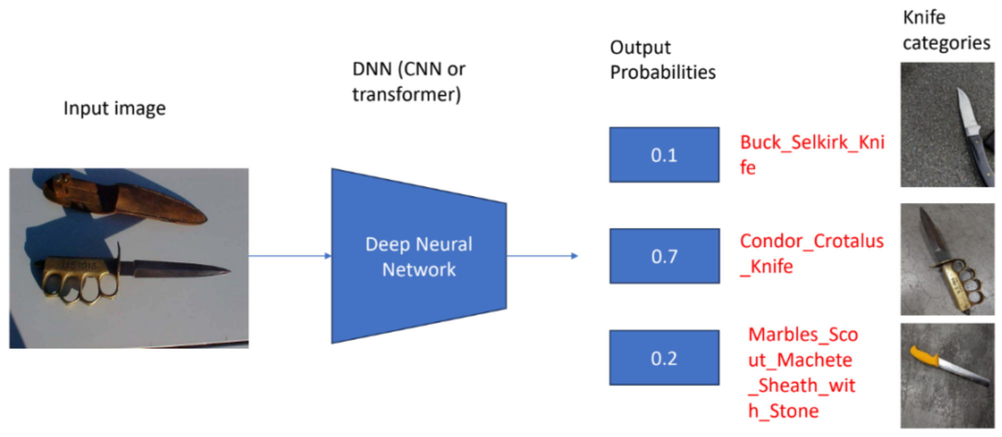


Figure 1: Overview of a knife classification system.

Instructions

In this assignment, you will develop a deep-learning-based system to classify knife images into different categories i.e., classes. You will gain real-world experience in designing, training, and evaluating a deep neural network for accurate, fine-grained knife classification.

You are required to document your findings in a report, divided into three sections as outlined later. Complete each section (in this document) according to the provided guidance.

Dataset

Use the "Knife classification" training dataset (provided here), which includes approximately 9,928 knife images categorised into 192 classes, along with a test dataset consisting of 351 images to complete the assignment. The performance of your model will be evaluated using the mean Average Precision (mAP) metric.

Important note:
Please do not distribute this dataset or use it for any purposes other than this assignment.

Compute platform

To complete this coursework assignment, you will need to use Google Colab. You are welcome to revisit the lab sheets you engaged with during this module for further guidance that can support you in completing this assignment.

Getting started

To prepare for this assignment, it is recommended that you engage with the following readings, available via the Reading List, before starting work on developing your knife classification system:

- "Fine-grained image analysis with deep learning: A survey" by Wei Xiu-Shen, Jianxin Wu, and Quan Cui
- "Weapon classification using deep convolutional neural network" by Neelam Dwivedi, Dushyant Kumar Singh, and Dharmender Singh Kushwaha

Additionally, you are encouraged to revisit the lectures in this module and the lab sessions on using Python and PyTorch to train and test CNNs for image classification. The PyTorch Introductory lab sheet and the lecture slides on these topics are available on SurreyLearn.

Support in programming

You can consult this GitHub repository for a reference code base. Should you experience any problems during the development of your knife classification, please feel free to open an issue.

Deliverables

For this assignment, you are required to run your project in the Colab. For the report, you must document the experiments performed using your software.

Additional instructions

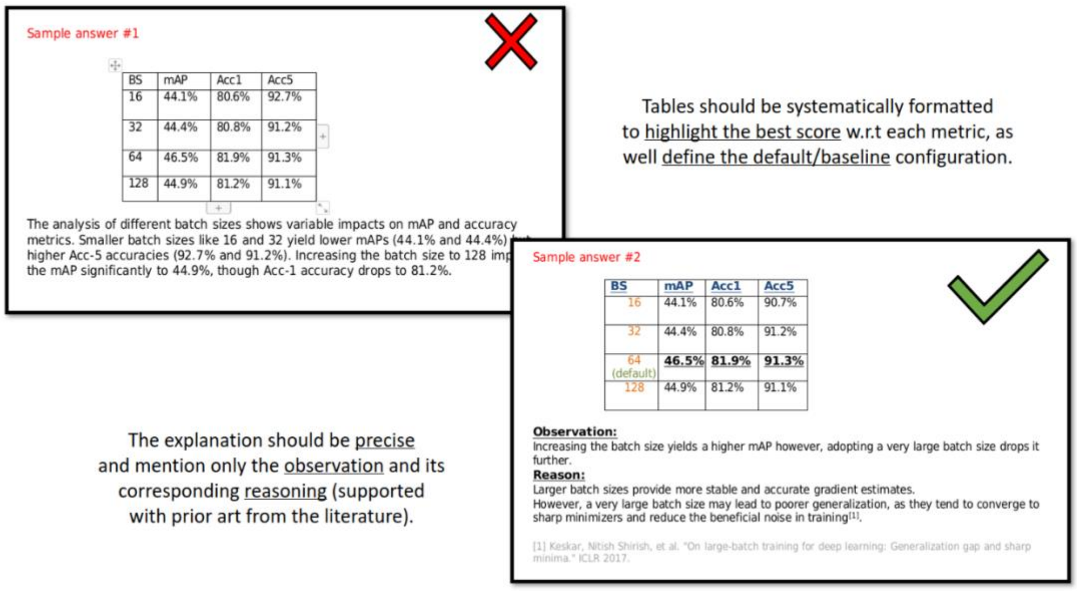
Keep the following guidance in mind as you work on producing the deliverables for this assignment:

Hyperparameters

For the parameters that are not learnable, you can set them before starting the training.

Format

- Ensure your responses to each question are on the spot as shown in Figure 2. Focus on the specific question and not move away.



Sample answer #1

BS	mAP	Acc1	Acc5
16	44.1%	80.6%	92.7%
32	44.4%	80.8%	91.2%
64	46.5%	81.9%	91.3%
128	44.9%	81.2%	91.1%

The analysis of different batch sizes shows variable impacts on mAP and accuracy metrics. Smaller batch sizes like 16 and 32 yield lower mAPs (44.1% and 44.4%) higher Acc-5 accuracies (92.7% and 91.2%). Increasing the batch size to 128 imp the mAP significantly to 44.9%, though Acc-1 accuracy drops to 81.2%.

Tables should be systematically formatted to highlight the best score w.r.t each metric, as well define the default/baseline configuration.

Sample answer #2

BS	mAP	Acc1	Acc5
16	44.1%	80.6%	90.7%
32	44.4%	80.8%	91.2%
64 (default)	**46.5%**	**81.9%**	**91.3%**
128	44.9%	81.2%	91.1%

Observation:
Increasing the batch size yields a higher mAP however, adopting a very large batch size drops it further.
Reason:
Larger batch sizes provide more stable and accurate gradient estimates.
However, a very large batch size may lead to poorer generalization, as they tend to converge to sharp minimizers and reduce the beneficial noise in training[1].

[1] Keskar, Nitish Shirish, et al. "On large-batch training for deep learning: Generalization gap and sharp minima." ICLR 2017.

The explanation should be precise and mention only the observation and its corresponding reasoning (supported with prior art from the literature).

Figure 2: Response example.

Log files

- Submit your own log files from your codebase. You are required to submit a separate log file for each experiment-based question.

- Do not change the log file structure. Each log file is watermarked and unique to each run.

- Name each log file according to the section number and question number: (log_sec{Section_num}_q{Question_num}.txt). For example, the log file generated for an experiment that corresponds to Section 1 Question 2, should be named as “log_sec1_q2.txt”.

Presentation and clarity

In addition to the technical content, the presentation and clarity of your writing will be assessed.

Model performance metrics

While different metrics are available, it is recommended that you prioritise mAP when selecting the best-performance models.

Start your assignment

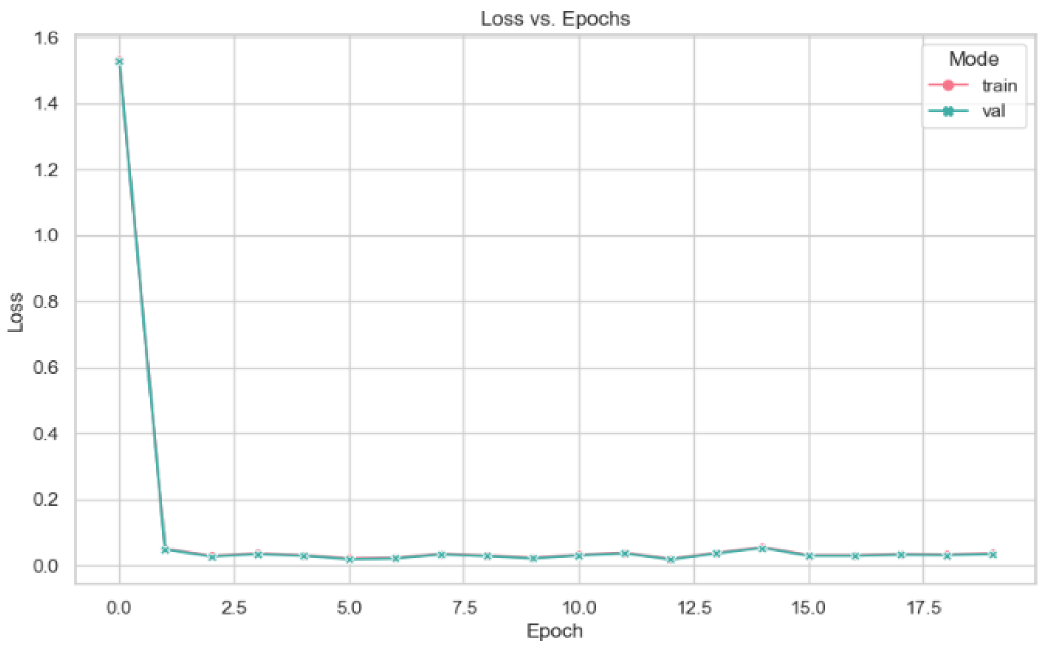
Write your assignment in the designated space provided below. Each of the sections outlines the tasks required and specifies the information you must include in your report for that section.

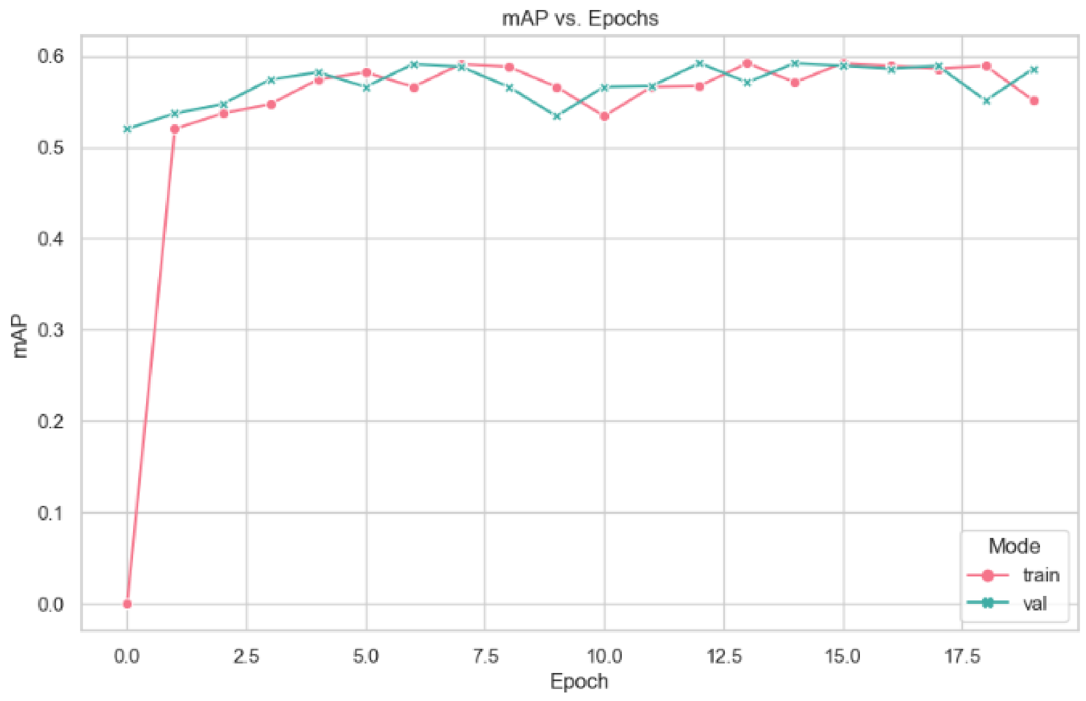
Section 1: Familiarity with the code provided (40 marks)

1. Run the code using the default settings. Discuss the steps involved in the training and evaluation process, and explain the impact of the observed performance using an appropriate metric. (20 marks)
2. Apply a different CNN variant from the one used in the code provided. Critically compare the results with those observed in Question 1. (10 marks)
3. Apply one more neural network architecture different from those in Questions 1 and 2. Critically discuss and compare the results from these previous two steps. (10 marks)

Note:
For Questions 2 and 3, ensure the new architecture is explicitly specified in your shell script command if you are using one from the codebase.

Solution: 1.1

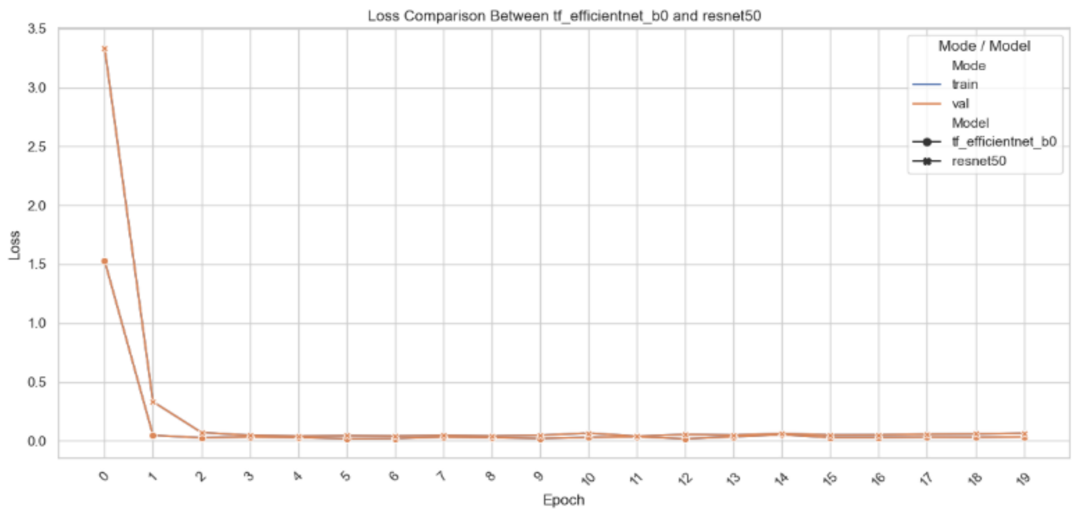


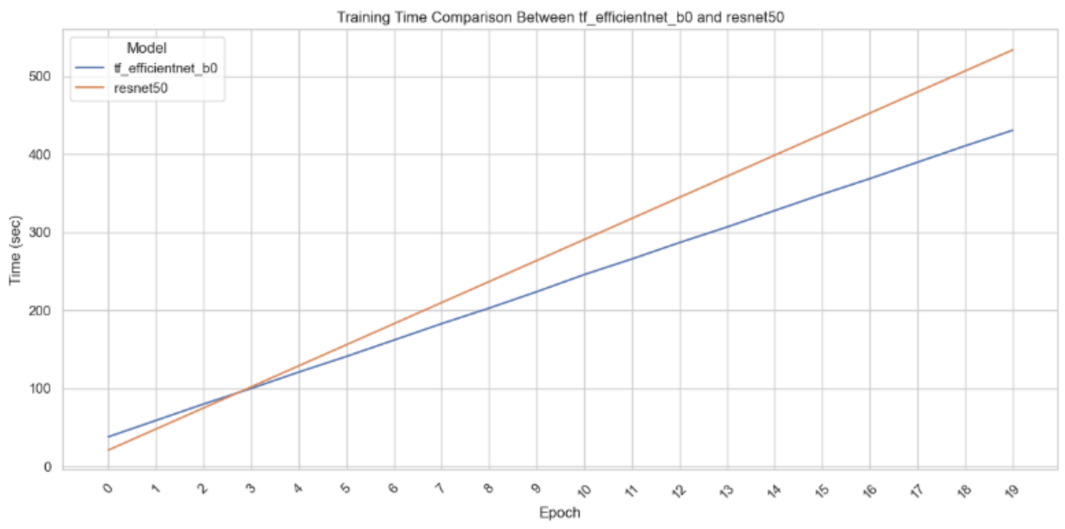


The user specifies model hyperparameters in **test.sh**, and the model is fetched from timm. Data preparation with augmentations and transformations occurs in **data.py**. In **main.py**, the training loop performs forward propagation, loss calculation with cross-entropy, and weight updates using the specified optimizer and scheduler. Evaluation computes the mAP, reflecting classification accuracy. Logs file created, provides a detailed record of performance.

Default model's performance: mAP improves from 0.000 to 0.592 by epoch 12, then stabilizes. The loss plot shows no overfitting or underfitting, as training and validation losses align closely throughout.

Solution: 1.2





Model Type	Best Epoch	Train Loss	Val Loss	Train mAP	Val mAP
tf_efficientnet_b0	12	0.017	0.017	0.567	0.592
resnet50	**4**	**0.041**	**0.041**	**0.677**	**0.69**

Observation:

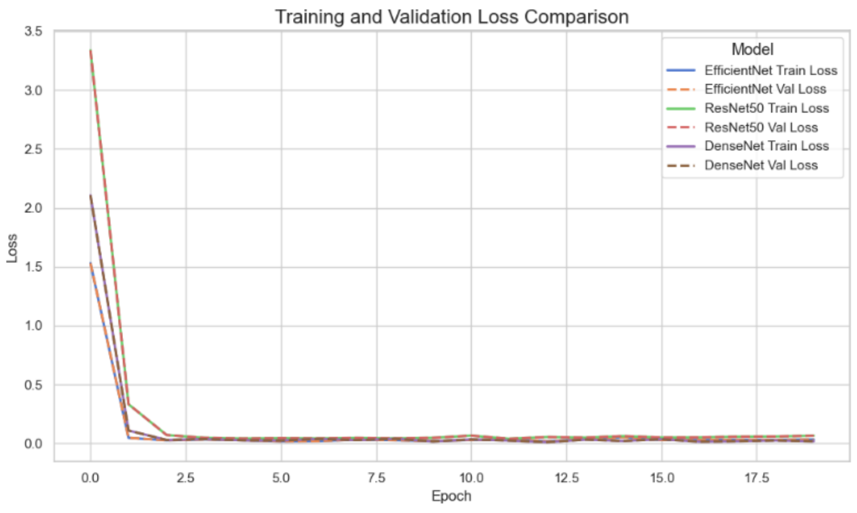
Switching from tf_efficientnet_b0 (Q1.1) to resnet50 increases validation mAP from 0.592 to 0.69 and achieves faster convergence (Best Epoch = 4 vs. 12) with higher training mAP (0.677 vs. 0.567). Loss plot infers that both models converge well, but Resnet50 linearly takes longer to train as epochs increase.

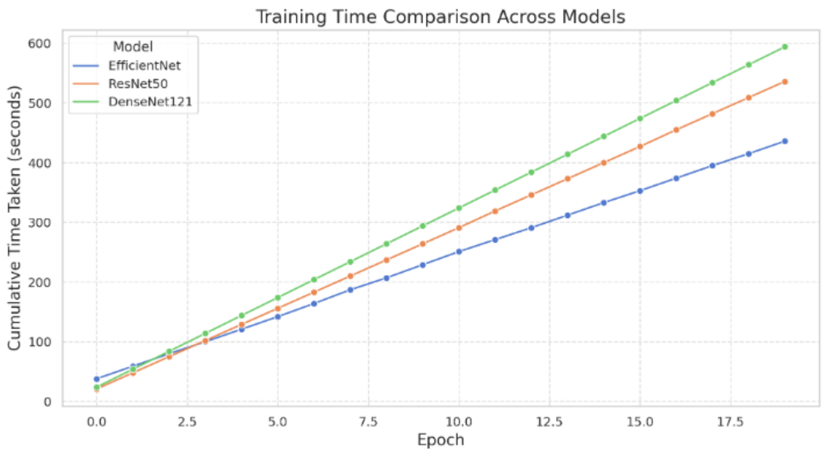
Reason:

- ResNet50's deep residual structure allows better feature extraction, leading to improved generalization [1].
- EfficientNet, using compound scaling to balance depth, width, and resolution, converges slower and has lower mAP performance [2] but is designed for accuracy and computational efficiency, resulting in faster training [4].

Solution: 1.3

Model Type	Best Epoch	Train Loss	Val Loss	Train mAP	Val mAP
tf_efficientnet_b0	12	0.017	0.017	0.567	0.592
resnet50	**4**	**0.041**	**0.041**	**0.677**	**0.69**
densenet121	7	0.028	0.028	0.593	0.604





Observation:
DenseNet121 achieves a validation mAP of 0.604 at epoch 7, higher than EfficientNet-B0 (0.592) but lower than ResNet50 (0.69). Its loss (0.028) is between ResNet50's (0.041) and EfficientNet-B0's (0.017).

Reason:

- As per line chart, ResNet50 performs best at most epochs due to its deep residual learning for complex feature extraction [1]
- DenseNet121's dense connections promote feature reuse, offering strong performance but less than ResNet50 and being more computationally demanding [3].
- EfficientNet-B0 prioritizes stability and efficiency, resulting in the lowest mAP but the fastest processing time [2].

[1] He, Kaiming, et al. "Deep residual learning for image recognition." CVPR 2016.
[2] Tan, Mingxing, and Quoc V. Le. "EfficientNet: Rethinking model scaling for convolutional neural networks." ICML 2019.
[3] Huang, Gao, et al. "Densely connected convolutional networks." CVPR 2017.
[4] EfficientNetV2: Smaller Models and Faster Training 2104.00298

Section 2: Dataset preparation and augmentation experiment (30 marks)

Begin with the default data augmentation setting of the default command line, which uses colour jittering.

1. Further append on top two additional data augmentation techniques, one at a time e.g., Default + "random rotation", Default + "horizontal flip". Compare the results with the default configuration in the provided code and discuss the differences in performance. (20 marks)

2. Combine the augmentation techniques from Question 1 to find the best-performing combination. Highlight any improvement or drop in the overall score. (10 marks)

Note: Resnet50, best performing model from Section 1 is used in Section 2

Solution: 2.1

Augmentation Type	Best Epoch	Train Loss	Val Loss	Train mAP	Val mAP
Default + Random Rotation (Val : 10)	2	0.085	0.085	0.642	0.682
Default + Horizontal Flip (Val : 10)	**2**	**0.072**	**0.072**	**0.661**	**0.71**
Default (resnet50)	4	0.041	0.041	0.677	0.69

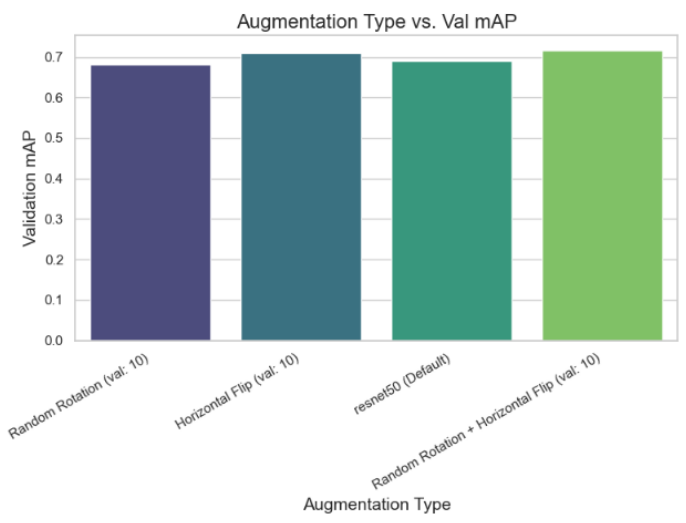
Observation:

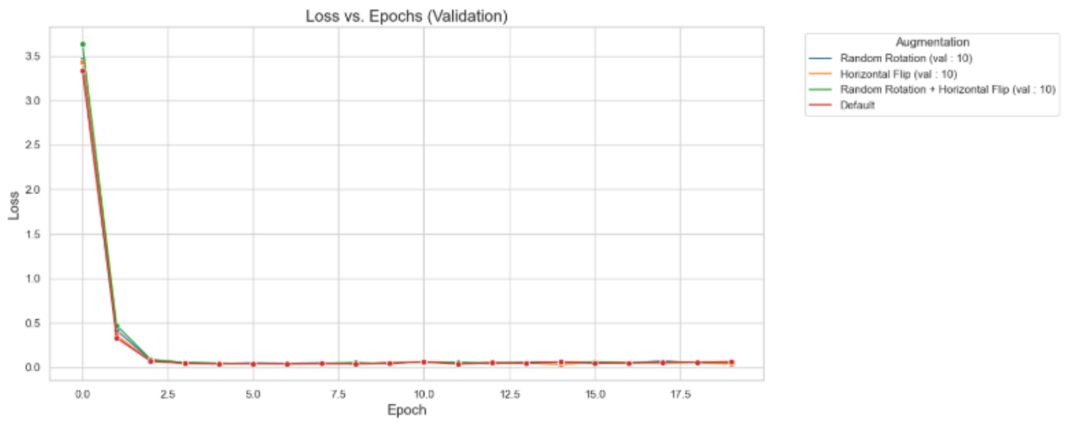
The "Default + Horizontal Flip" configuration has the highest validation mAP of 0.71, which is better than the default configuration

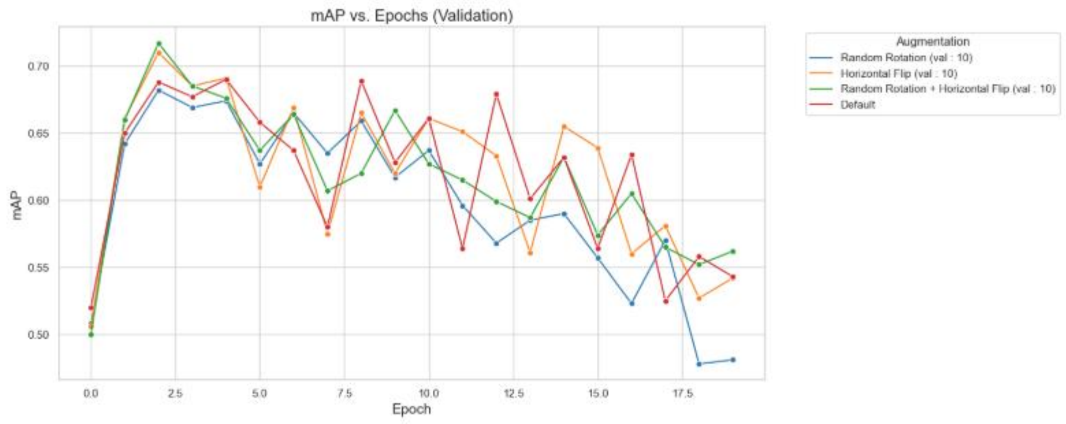
Reason:

- Horizontal flipping makes the model more robust by teaching it to recognize objects regardless of their left-right orientation [1]
- Simple transformations like flipping can sometimes be more effective than more complex ones like rotation. This is consistent with research findings [2]

Solution: 2.2







Augmentation Type	Best Epoch	Train Loss	Val Loss	Train mAP	Val mAP
Random Rotation (Val : 10)	2	0.085	0.085	0.642	0.682
Horizontal Flip (Val : 10)	2	0.072	0.072	0.661	0.71
resnet50 (Default)	4	0.041	0.041	0.677	0.69
Random Rotation + Horizontal Flip (Val : 10)	**2**	**0.091**	**0.091**	**0.66**	**0.717**

Observation:

Combining the "Random Rotation" and "Horizontal Flip" data augmentation techniques yields the best overall performance, with a validation mAP of 0.717

Reason:

Applying both rotation and horizontal flip together creates more diverse and representative training data, which helps the model learn more robust and generalizable features. As seen in research, the combination of augmentation techniques helps models learn more robust features [3]

References:

[1] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in neural information processing systems, 25, 1097-1105.

[2] Sato, I., Nishimura, H., & Yokoi, K. (2015). APAC: Augmented pattern classification with neural networks. arXiv preprint arXiv:1505.03229

[3] Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., & Le, Q. V. (2019). AutoAugment: Learning augmentation strategies from data. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 113-123).

Section 3: Exploration of Hyperparameters (20 marks)

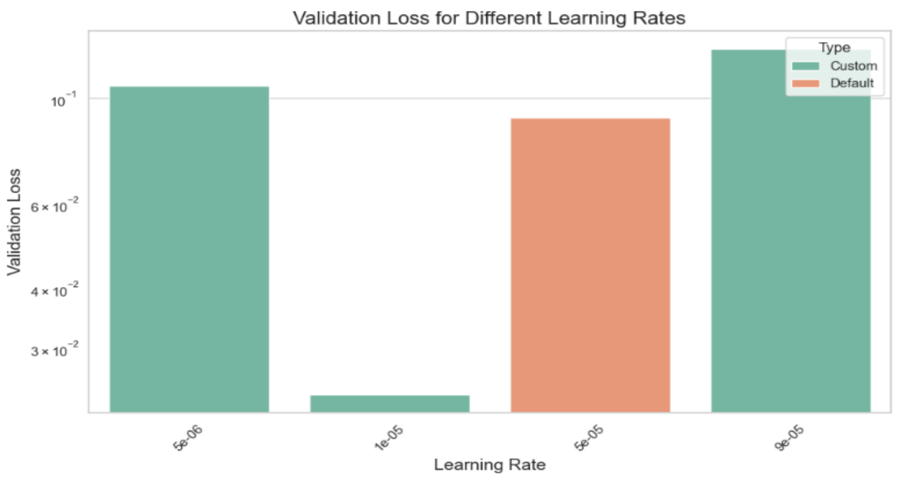
Start with the default learning rate (LR) and batch size (BS).

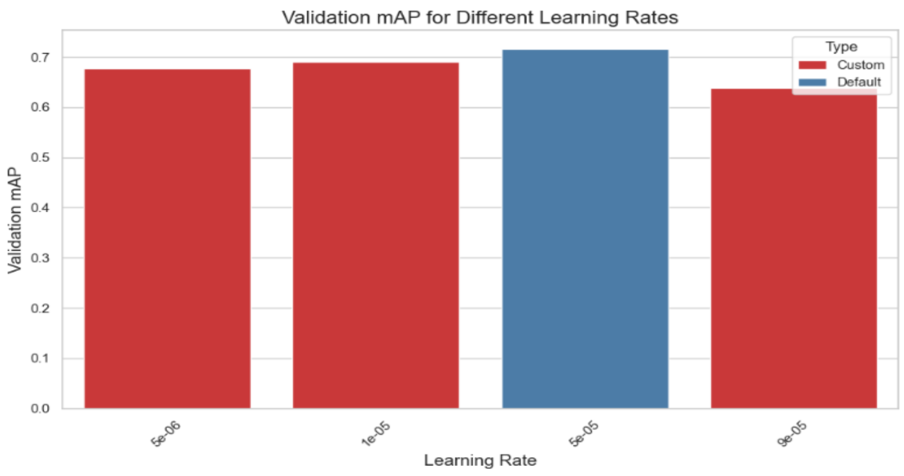
1. Exploration of LR. (10 marks)
    a. Experiment with three values of LR (in addition to the default value).
    b. Discuss the observed impact of each value on overall performance.

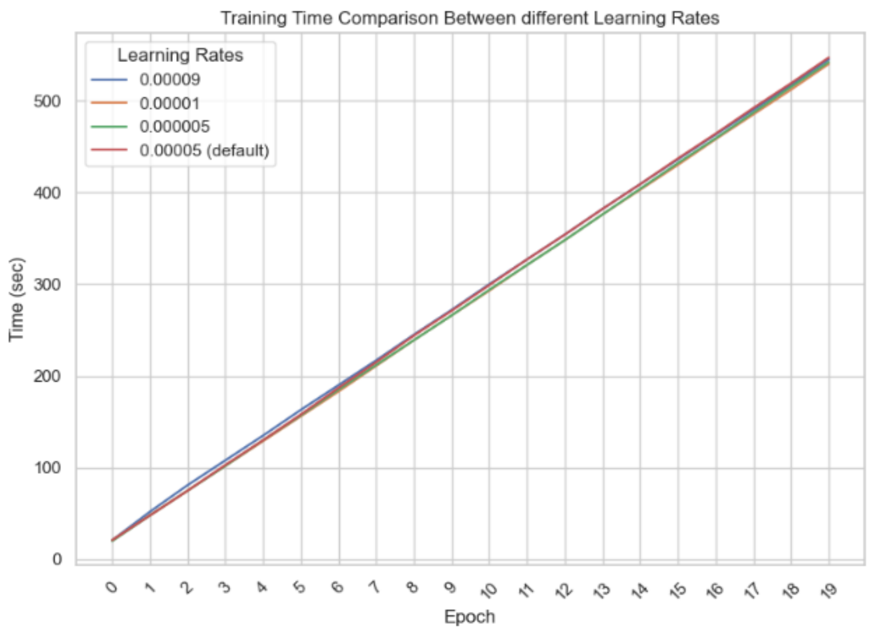
2. Exploration BS. (10 marks)
    a. Using the best LR value from the experiments in Question 1, test three different values of the BS in addition to the default value.
    b. Discuss how the changes in BS affect overall performance.

Note: ResNet50, the best performing model from Section 1, combined with the augmentation of random rotation and horizontal flip (val: 10) from Section 2, is used in Section 3.

Solution: 3.1







Learning Rate	Best Epoch	Train Loss	Val Loss	Train mAP	Val mAP
0.00009	1	0.127	0.127	0.604	0.638
0.00001	19	0.024	0.024	0.685	0.691
0.000005	18	0.106	0.106	0.66	0.677
0.00005 (Default)	2	0.091	0.091	0.66	0.717

Observation:

- The default learning rate achieves the highest validation mAP of 0.717 and training time across different LR is almost similar.
- Both the highest (0.00009) and lowest (0.000005) learning rates result in suboptimal validation mAP.

Reason:

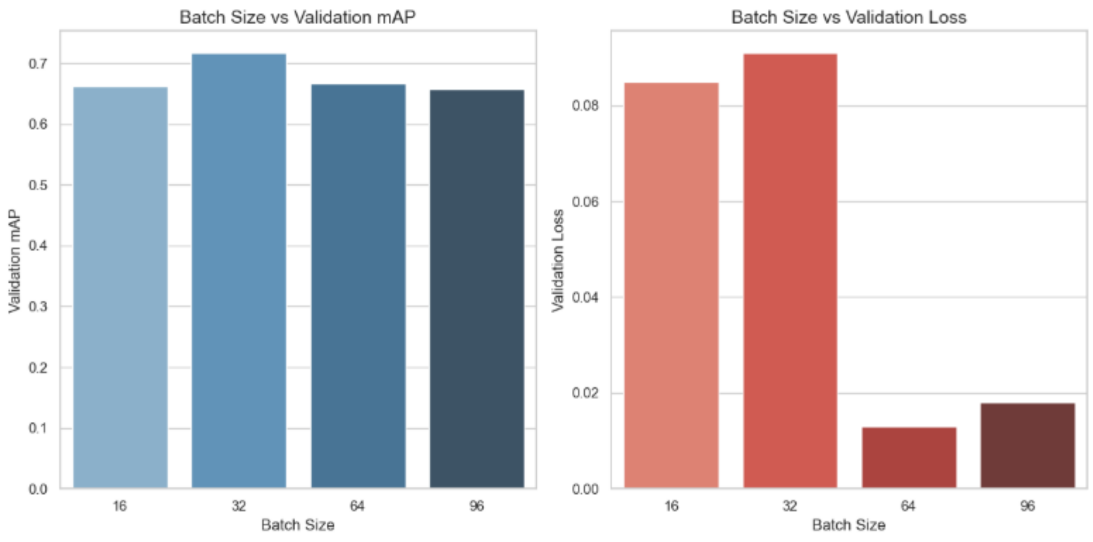
- Lower learning rates (0.000005) struggle to escape local minima, leading to poor performance [1]
- while higher learning rates (0.00009) cause aggressive parameter updates, leading to instability and poor convergence [2].

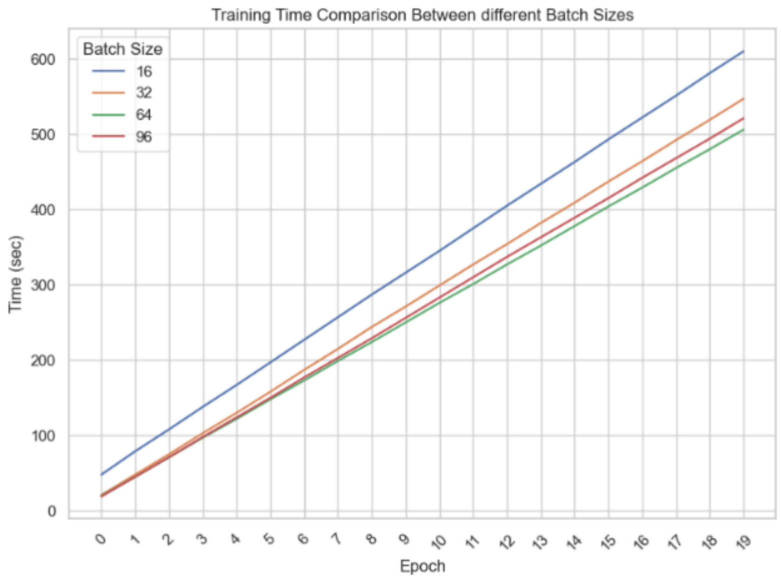
References:

[1] Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013). On the importance of initialization and momentum in deep learning. In International conference on machine learning (pp. 1139-1147). PMLR.
[2] Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Solution: 3.2

Note: Since from the previous section, we were unable to find any learning rate whose validation mAP was greater than 0.717, we have therefore gone with the default learning rate of 0.00005 for this section.





Batch Size	Best Epoch	Train Loss	Val Loss	Train mAP	Val mAP
16	2	0.085	0.085	0.647	0.662
64	12	0.013	0.013	0.629	0.666
96	14	0.018	0.018	0.64	0.657
32 (Default)	2	0.091	0.091	0.66	0.717

Observation

- The default batch size of 32 achieves the highest validation mAP of 0.717
- A smaller batch size of 16 results in a validation mAP of 0.662, which is lower than the default and also slowest to train.

- Larger batch sizes of 64 and 96 has lower training and validation loss but that still it leads to even lower validation mAPs of 0.666 and 0.657 respectively.

Reasons

- smaller batches (16) can lead to noisier gradients and slower convergence [1]
- Larger batch sizes like 64 and 96 may cause the model to miss important details in the data, leading to underfitting and lower generalization [2]

Reference:

[1] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. In International Conference on Learning Representations.

[2] Hoffer, E., Hubara, I., & Soudry, D. (2017). Train longer, generalize better: closing the generalization gap in large batch training of neural networks. Advances in Neural Information Processing Systems, 30.

Rubric

Section 1: Familiarity with the code provided (40 marks)

Task 1.1: Running and understanding the code (20 marks)

- **Understanding the code execution**
  - *Criteria*: Clear understanding and explanation of the code flow and processes.
  - *Indicators*: Detailed description of data loading, model initialisation, training loop, and evaluation steps.

(5 marks)

- **Metric explanation and performance analysis**
  - *Criteria*: Appropriate choice and understanding of the evaluation metric(s) (e.g., accuracy, mAP).
  - *Indicators*: Correct calculation and interpretation of the metric values. Discussion on what these values imply about the model's performance.

(10 marks)

- **Discussion of observed performance**
  - *Criteria*: Insightful discussion on the observed results.
  - *Indicators*: Analysis of performance patterns, potential overfitting/underfitting issues, and suggestions for improvement.

(5 marks)

Task 1.2: Applying a different CNN variant (10 marks)

- **Implementation of a new CNN variant**
  - *Criteria*: Correctly modifying the code to implement a different CNN architecture.
  - *Indicators*: Model architecture is different from the provided one, and code runs without errors.

(4 marks)

- **Comparison and critical analysis**
  - *Criteria*: Critical comparison between the original and new CNN results.
  - *Indicators*: Use of quantitative results to support comparison, discussion on potential reasons for performance differences.

(6 marks)

Task 1.3: Applying another neural network architecture (10 marks)

- **Implementation of a new neural network**
    - *Criteria*: Correctly implementing a neural network architecture different from those in Tasks 1.1 and 1.2.
    - *Indicators*: Model architecture is distinct, and code runs correctly.

(4 marks)

- **Comprehensive comparison and analysis**
    - *Criteria*: Deep comparative analysis between all three architectures.
    - *Indicators*: Insightful discussion on the strengths and weaknesses of each model, based on quantitative results and qualitative observations.

(6 marks)

Section 2: Dataset preparation and augmentation experiment (30 marks)

Task 2.1: Augmentation techniques (20 marks)

- **Implementation of additional augmentations**
    - *Criteria*: Successful addition of two new augmentation techniques.
    - *Indicators*: Correct application and explanation of chosen techniques, code runs without errors.

(8 marks)

- **Performance comparison and analysis**
    - *Criteria*: Comparative analysis of the impact of each augmentation setting.
    - *Indicators*: Clear explanation of performance differences between default and new settings, supported by quantitative results.

(8 marks)

- **Discussion on results and insights**
    - *Criteria*: Insightful interpretation of how augmentations affected model performance.
    - *Indicators*: Discussion on potential reasons for performance changes and implications for model robustness.

(4 marks)

Task 2.2: Best-performing augmentation combination (10 marks)

- **Combining techniques and experimentation**
    - *Criteria*: Correct implementation of combined augmentation techniques.

- *Indicators*: Code successfully combines the techniques and executes without errors.

(5 marks)

- **Analysis of improvement or decline**
    - *Criteria*: Detailed discussion on whether the combined augmentation improved performance.
    - *Indicators*: Use of appropriate metrics to highlight any performance change, supported by logical reasoning.

(5 marks)

Section 3: Exploration of hyperparameters (20 marks)

Task 3.1: Learning rate exploration (10 marks)

- **Experimentation with different LR values**
    - *Criteria*: Correct experimentation with three additional LR values.
    - *Indicators*: Clear documentation of the chosen LR values and implementation.

(4 marks)

- **Performance analysis of each LR**
    - *Criteria*: Thorough analysis of the effect of each LR value on performance.
    - *Indicators*: Use of metrics to compare performance across LR values, discussion on the reasons behind observed changes.

(6 marks)

Task 3.2: Batch size exploration (10 marks)

- **Experimentation with different BS values**
    - *Criteria*: Correct experimentation with three additional BS values.
    - *Indicators*: Clear documentation of the chosen BS values and implementation using the best-performing LR from Task 3.1.

(4 marks)

- **Performance analysis of each BS**
    - *Criteria*: Detailed analysis of the impact of each BS value on performance.
    - *Indicators*: Use of metrics to compare performance across BS values, discussion on the reasons behind observed changes.

(6 marks)

Report writing and presentation (10 marks)

- **Figures should be well-presented, with readable axes and labels.**

(2 marks)

- **The writing should be clear, grammatically correct, and easy to follow.**

(6 marks)

- **Tables are used when discussing results involving multiple hyperparameters values**
    - *Criteria*: Clarity, structure, and depth of the written report.
    - *Indicators*: Logical flow, clear explanation of methods and results, proper use of figures/tables to support the analysis.

(2 marks)

Total: 100 marks