Module	Advanced Topics in Computer Vision and Deep Learning Coursework Assignment: EEEM071
Assessment type	Coursework assignment
Weighting	25%
Deadline	4:00PM, Friday, 11 April 2025 (Week 10)
STUDENT URN	6891988
STUDENT Name	Frank Tsu-Fang Lu

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.

Vehicle Re-identification

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 lab sessions and 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: EEEM071-[insert your URN number]-Assignment e.g., EEEM071-1234567-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

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 1. Focus on the specific question and not move away.

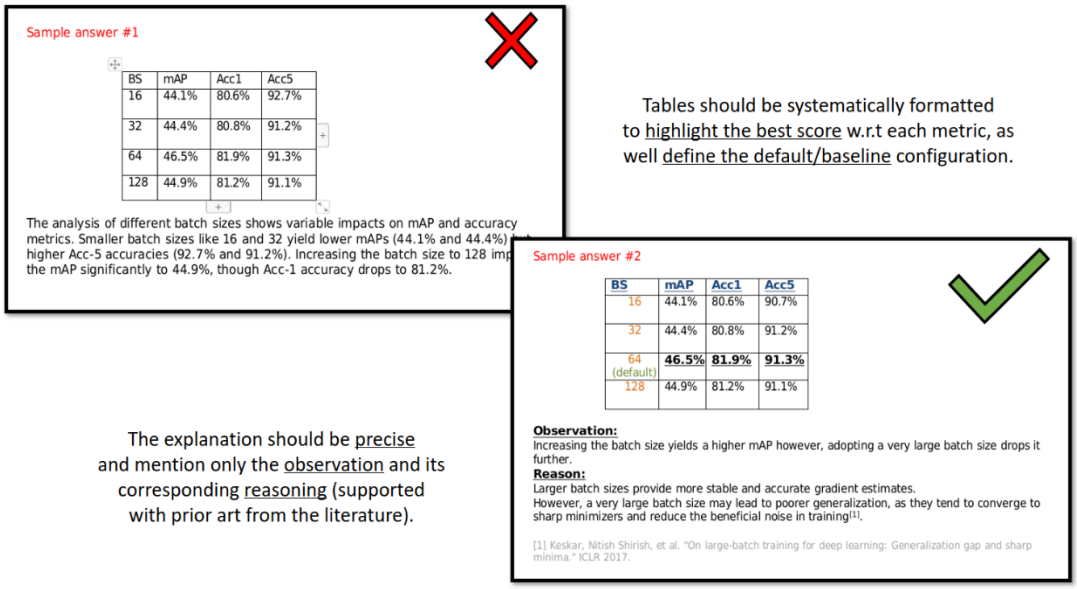


Figure 1: Response example.

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 (30 marks)

1. Run the code using the default settings. Discuss the training and evaluation process (*mention the loss functions used*) followed by implications of the observed performance using an appropriate metric. (10 marks)

2. Apply another CNN variant (that is not provided in the default settings). Critically discuss and contrast the results with what observed in question 1 above. (10 marks)

3. Apply one more neural network architecture not from the same family as the above questions. Critically discuss and contrast the results with what observed questions 1 and 2 above. (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.

Start writing here:

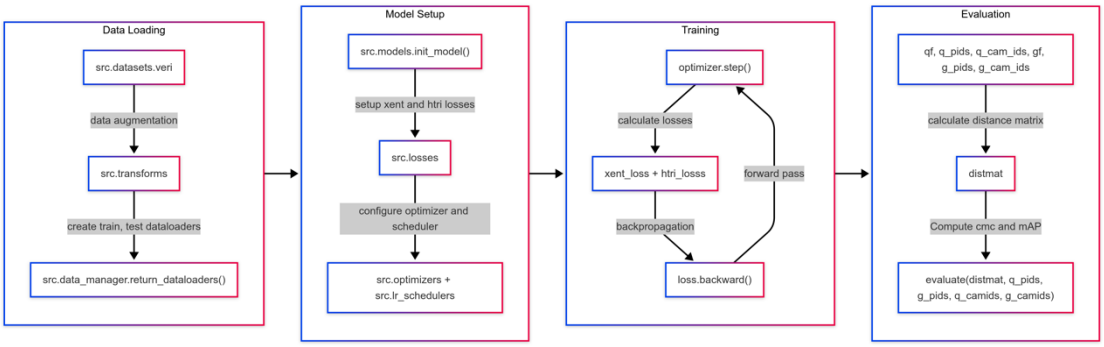


Figure 1: Codebase visualized: Data loading via ImageDataManager class → Model Setup → Training with a combined xent and htri loss → Evaluation with CMC and mAP

$$H(P^* \mid P) = -\sum_i \underbrace{P^*(i)}_{\text{TRUE CLASS DISTIRBUTION}} \log \underbrace{P(i)}_{\text{PREDICTED CLASS DISTIRBUTION}}$$

Equation 1: The cross-entropy loss function measures the difference between the predicted class

probabilities and the true class probabilities.

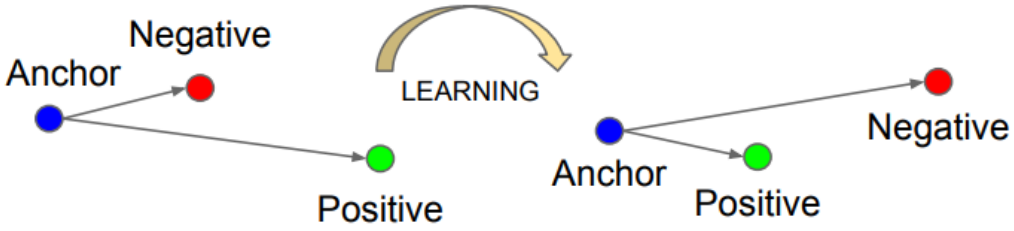


Figure 2: Triplet loss pulls an anchor closer to a positive (same vehicle) while pushing it away from a negative (different vehicle) [1]

1. **Observation:**
   The training process uses cross-entropy loss to guide classification and hard triplet loss to cluster similar vehicles [1]. The evaluation uses mAP to measure overall retrieval quality and CMC/ rank-1 accuracy for top match performance. MobileNet demonstrates a relatively high rank-1 accuracy (**80.10%)** with an impressive runtime of only **10m 42s**, but low mAP (**44.72%).** The gap between these metrics and the absence of an "elbow" in the total loss curve after 10 epochs suggests the model is underfitting.

   Reason:
   We prioritize mAP as the primary metric because it evaluates the entire ranking list considering precision and recall, not just the top-k matches. In re-identification, each query vehicle may have multiple correct matches, and CMC only considers whether at least one correct match appears at a certain rank (rank-1). MobileNet's high rank-1 accuracy verifies its effective Squeeze-and-Excitation modules enhancing channel dependencies [2]. However, the low mAP suggests the model captures primary visual features well but has limited invariance to viewpoint and lighting changes across camera views [3].

Built-in Architecture	mAP (%)	Rank-1 (%)	Runtime
MobileNet_V3_Small (Default)	44.72	80.10	**10m 42s**
VGG16	**20.99**	**56.62**	**30m 48s**
ResNet18	51.34	84.09	12m 51s
ResNet18_FC512	52.75	86.77	13m 13s
ResNet34	48.69	82.18	15m 34s
ResNet34_FC512	50.15	84.15	16m 36s
ResNet50	53.37	84.86	21m 33s
ResNet50_FC512	**54.54**	**87.01**	24m 15s

Table 1: All experiments are performed with the 10-epoch limits and a default learning rate of 3e-03 and a batch size of 64. Please see `train.sh` for other default parameters.

2. **Observation**
VGG's metrics substantially dropped (**20.99% mAP and 56.62% Rank-1**) compared to MobileNet, with triple the runtime (**30m 48s**).

Reason
This is explained by VGG's uniform down sampling strategy, causing a progressive loss in fine-grained low-level details crucial for re-identification. Unlike MobileNet's use of SE modules, VGG rigidly process all features equally, limiting its discriminative capabilities. Finally, the deeply stacked convolutional layer tripled the computational cost despite its inferior performance [4]. This goes to prove that **deeper is not better, if done trivially**.

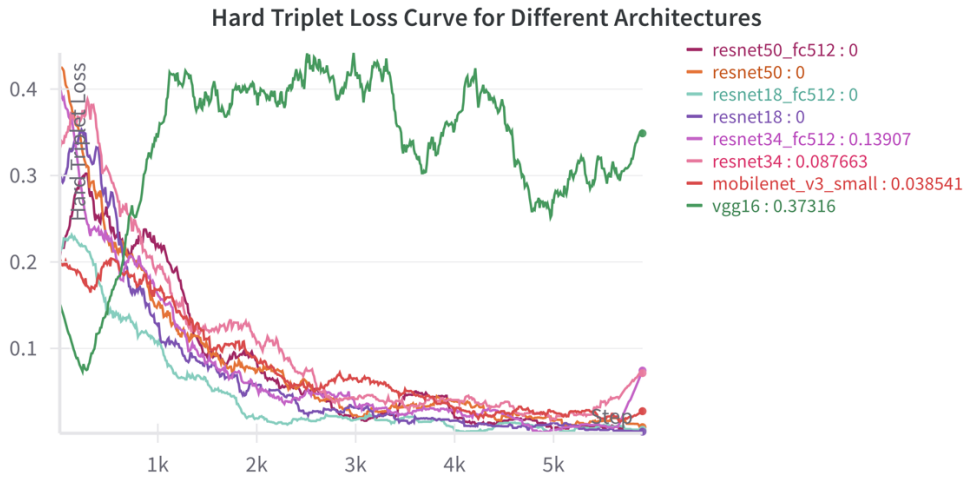


Figure 3: ResNet variants successfully creates discriminative feature embeddings (0 htri-loss by end of training) while VGG16 struggles with unstable triplet loss.

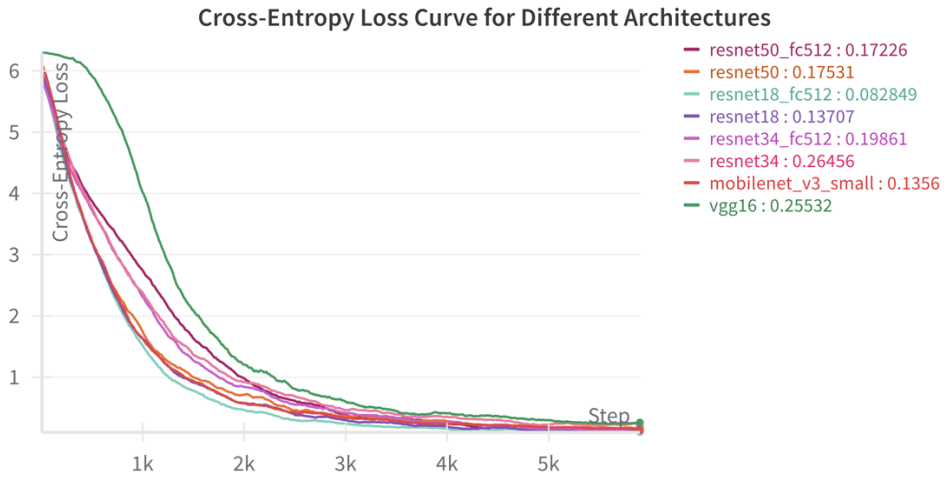


Figure 4: ResNet variants converge to similarly low values (~0.1), while VGG16 maintains consistently higher loss throughout, explaining its poorer performance. MobileNet's convergence remains competitive despite its lightweight design.

3. **Observation**
ResNet50_FC512 achieves the highest mAP (**54.54%)** and Rank-1 (**87.01%**), outperforming both MobileNet and VGG16 significantly.

Reason

This is due to ResNet50's residual connections, which address vanishing gradients and preserve low-level information flow across layers. FC512 also introduces two enhancements: a reduced last stride (2 → 1) that preserves spatial information in the final feature maps, and a 512-dimension bottleneck layer before classification. This dimensionality reduction (2048 → 512) creates a more discriminative and generalizable embedding space compared to direct class mapping [5].

Custom Architecture	mAP (%)	Rank-1 (%)	Runtime
ResNet18 with SE	50.55	84.15	13m 46s
ResNet18_FC512 with SE	**53.31**	**85.04**	**15m**
ResNet50 with SE	49.75	81.05	27m 36s
ResNet50_FC512 with SE	53.11	85.82	28m 45s
ResNet18 with SA	27.96	55.54	**56m 59s**
vit_b_16	**21.87**	**48.81**	45m 46s

Table 2: Custom Implementations of ResNet with Squeeze-and-Excitation [6], Self-Attention ResNet with Self-Attention [7] and a small Vision Transformer model [8]. Due to longer training durations (exceeding 10 epochs) and the need for distinct hyperparameter tuning and potentially a lot more training samples for convergence [8], I have excluded these models from the experiments. See `log_sec1_extra.txt`.

Section 2: Dataset preparation and augmentation experiment (25 marks)

Begin with the default data augmentation setting with random, horizontal flip and Random2DTranslation.

1. Further append on top two additional data augmentation techniques, one at a time e.g., Default + "crop", Default + "horizontal flip", and Default + "blurring". 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. (5 marks)

Start writing here:

1. **Observation**
   The default augmentation (Random2DTranslation and horizontal flipping) already has promising results (**54.54% mAP**). Custom random perspective on ResNet18 Model slightly improves results **(+0.07% mAP)** while colour-based augmentations decrease performance consistently, with Colour Jitter having the largest drop **(-3.31% mAP).**

   Reason
   Geometric augmentations like flipping and translation enhance viewpoint invariance and leverage vehicle's bilateral symmetry**.** This is consistent with findings from single-modality image-based re-id in the survey paper [9]. Random2DTranslation's incorporation of probabilistic random cropping also improve robustness to occlusion [10]. Colour-based augmentations are the most detrimental. Unlike person re-identification where subjects may change clothing, vehicle re-identification relies heavily on colour as a primary identifying characteristic [3].

Built-In Augmentation	mAP (%)	Rank-1 (%)	Difference from Default (mAP)	Difference from Default (Rank-1)
No Augmentation	52.76	85.10	-1.78	-1.91
Default	**54.54**	**87.01**	**-**	**-**
Default + Random Erasing	53.58	84.86	-0.96	-2.15
Default + Colour Jitter	**51.23**	**84.80**	**-3.31**	**2.21**
Default + Colour Augmentation	51.88	84.92	-2.26	-2.09

Table 3: Augmentations carried out using the ResNet50_FC512 models.

Custom Augmentation	Model	mAP (%)	Rank-1 (%)	Difference from Default (mAP)	Difference from Default (Rank-1)
Default + Random Perspective	ResNet50 _FC512	53.10	84.68	-1.44	-2.33
Default + Random Perspective	**ResNet18 _FC512**	**54.61**	**87.07**	**+0.07**	**+0.06**

Table 4: Custom Random Perspective implementation using the ResNet18_FC512 models.

2. **Observation**

All augmentation combinations underperform compared to the default. The best combination is Default + Random Erasing + Colour Jitter + Colour Augmentation (**53.93% mAP, 85.04% Rank-1**), which still shows -**0.61% in mAP** and -**1.97% in Rank-1**. Notably, any combinations with colour-based augmentations consistently degrade model performance.

Reason

Using inappropriate augmentation potentially distorts crucial vehicle identification features. However, random erasing seems to partially offset the negative effects of colour-based augmentations [3]. This suggests occlusion simulation may help the model rely less on potentially misleading colour information. Overall, while preserving colour fidelity is important, carefully balancing augmentation diversity can also help maintain reasonable performance.

Augmentation	mAP (%)	Rank-1 (%)	Difference from Default (mAP)	Difference from Default (Rank-1)
Default + Random Erasing + Colour Jitter	53.08	84.86	-1.46	-2.15
Default + Random Erasing + Colour Augmentation	53.37	**84.68**	-1.17	**-2.33**
Default + Colour Jitter + Colour Augmentation	**52.35**	86.29	**-2.24**	-0.72
Default + Random Erasing + Colour Jitter + Colour Augmentation	**53.93**	**85.04**	**-0.61**	**-1.97**

Table 5: Combining different data augmentations.

Section 3: Exploration of Hyperparameters (25 marks)

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

1. Exploration of Learning Rate (LR). (10 marks)
    a. Experiment with 4 values of LR (in addition to the default value).
    b. Discuss the observed impact of each value on overall performance.
2. Exploration Batch sizes. (10 marks)
    a. Fixing the best LR value from the experiments in question 1 above, test 4 different values of the BS in addition to the default value.
    b. Discuss the impact observed on overall performance.
3. Exploration of the optimizer. (5 marks)
    a. Fixing the best Learning Rate value and best Batch Size value from the experiments in Questions 1 and 2, respectively, test with changing the optimizer to SGD, using PyTorch's internal class. Discuss the impact observed on overall performance.

Start writing here:

1. **Observation**
   Smaller learning rates (3e-05, 1e-04) outperform the default, while larger ones (3e-03, 1e-03, 1e-02) substantially degrade performance. A learning rate of 1e-04 achieves the highest performance (**64.96 mAP and 91.06 Rank-1),** a substantial **+11.55% mAP** and **+5.18% mAP** improvement respectively. Additionally, Higher rates show a jagged loss curve.

   Reason
   The default rate is (3e-04) is too aggressive for this task, causing the model to overshoot optimal solutions. Jagged loss curves at higher rates suggest unstable gradients, potentially leading to exploding gradients [11]. Smaller learning rates allow for more precise weight updates and hence better generalization and more stable convergence. However, the model may fail to escape suboptimal solutions and cause vanishing updates.

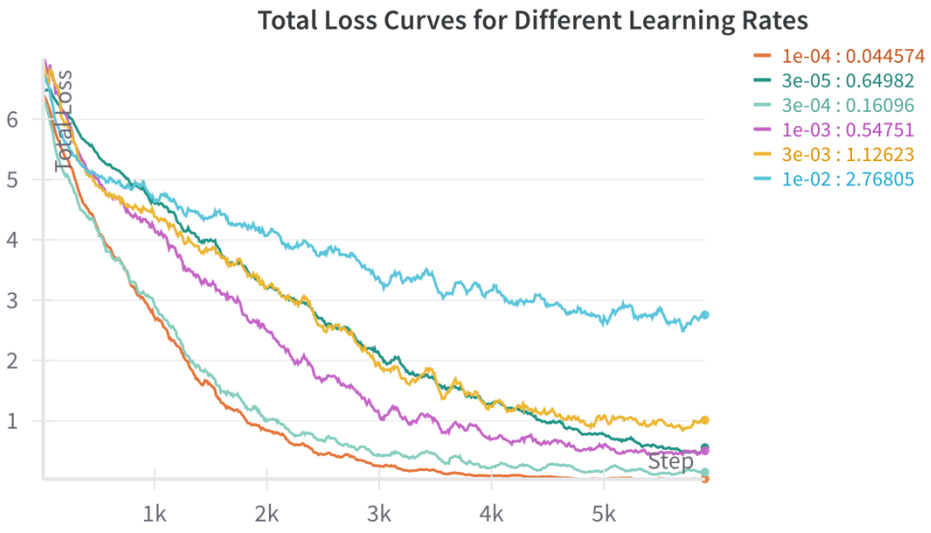


Figure 5: Total Loss Curves for Different Learning Rates

Learning Rate	mAP (%)	Rank-1 (%)	Difference from Default (mAP)	Difference from Default (Rank-1)
3e-05	63.76	89.63	+10.35	+1.43
1e-04	**64.96**	**91.06**	**+11.55**	**+5.18**
3e-04 (Default)	53.41	85.88	-	-
3e-03	28.90	65.20	-24.51	-20.68
1e-03	37.49	73.54	-27.47	-12.34
1e-02	**15.58**	**36.05**	**-37.83**	**-49.83**

Table 6: Tuning learning rate for ResNet50_FC512.

2. **Observation**
Small batch sizes (8, 16, 32) have a noticeable drop in mAP **(-8.45%, -7.61% and -4.7% respectively**). Larger batch sizes show minimal degradation **(-1.64% and -1.43%)** and converge the fastest (**<10k steps**). The loss curves also show that smaller batch sizes converge more slowly with higher volatility. A batch size of 8 has extremely jagged fluctuations for **40K+ steps**. A batch size of 64 is the most optimal with the lowest final loss (**0.045**).

Reason
High variance in gradient estimates is prominent in small batch training [12]. Smaller batch sizes result in noisier gradient estimates due to fewer samples per update, which results in less informative triplet combinations. Larger batch size increases the number of triplets, reducing gradient variance and leading to a more stable and consistent gradient update. However, this comes with the trade-off of less frequent weight updates.

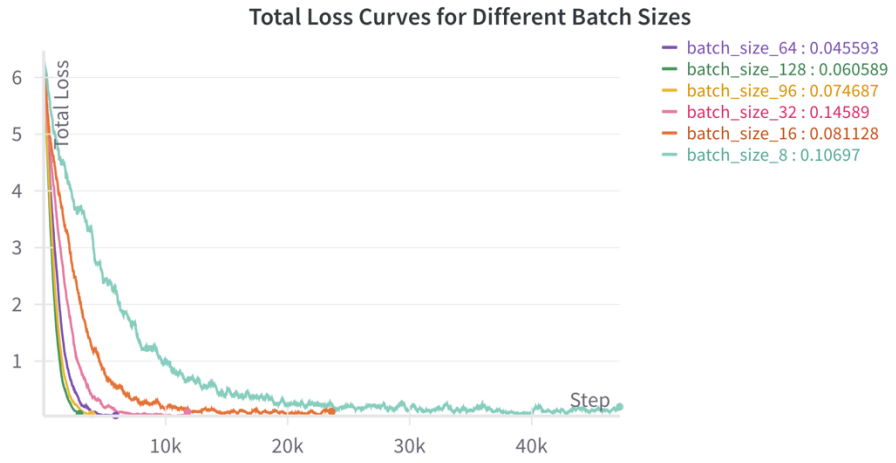


Figure 6: Total loss curves for different batch sizes

Batch Size	mAP (%)	Rank-1 (%)	Difference from Default (mAP)	Difference from Default (Rank-1)
8	**57.40**	88.56	**-8.45**	-3.39
16	58.24	**88.20**	-7.61	**-3.75**
32	60.99	88.86	-4.86	-3.09
64 (default)	**65.85**	**91.95**	**-**	**-**
96	64.21	90.58	-1.64	-1.37
128	64.42	90.82	-1.43	-1.13

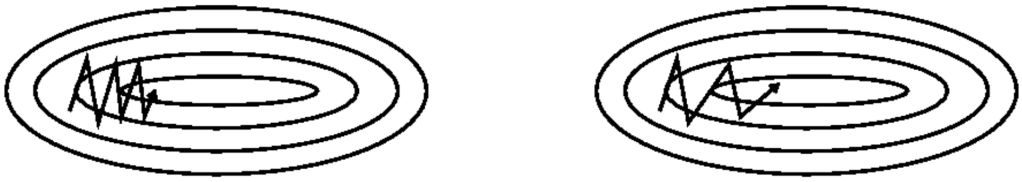
Table 7: Tuning batch size for ResNet50_FC512 while fixing learning rate to 1e04.

3. **Observation**
The default AMSGrad outperforms all alternatives with **66.94% mAP** and **91.36% Rank-1**. This is followed by Adam (**56.55% mAP and 87.07% Rank-1**). RMSProp and SGD both performed significantly worse with high gradient noise.

Reason
SGD performs poorly due to its uniform learning rate regardless of feature importance and its stochasticity generating noisy gradient updates [13]. RMSProp improves upon SGD by adapting learning rates per parameter but fails to handle sparse gradients. Adam introduces momentum (moving average of gradients) for smoother updates [13]. Finally, AMSGrad prevents rapid learning rate decay by maintaining the maximum of past squared gradients rather than using their exponential average [14]. This is helpful for vehicle re-identification where key features can appear sporadically across different camera views [3].

Optimizer	mAP (%)	Rank-1 (%)	Difference from Default (mAP)	Difference from Default (Rank-1)
SGD	**28.17**	**57.87**	**-38.77**	**-33.49**
RMSProp	32.37	69.73	-34.6	-21.63
AMSGrad (default)	**66.94**	**91.36**	**-**	**-**
Adam	56.55	87.07	-10.39	-4.29

Table 8: Tuning optimizer for ResNet50_FC512, with learning rate fixed to 1e04 and batch size fixed to 6



(a) SGD without momentum (b) SGD with momentum

Figure 7: Momentum helps accelerate SGD in the relevant directions across ravines (surface curves that are steeper in one dimension that in another) [13]

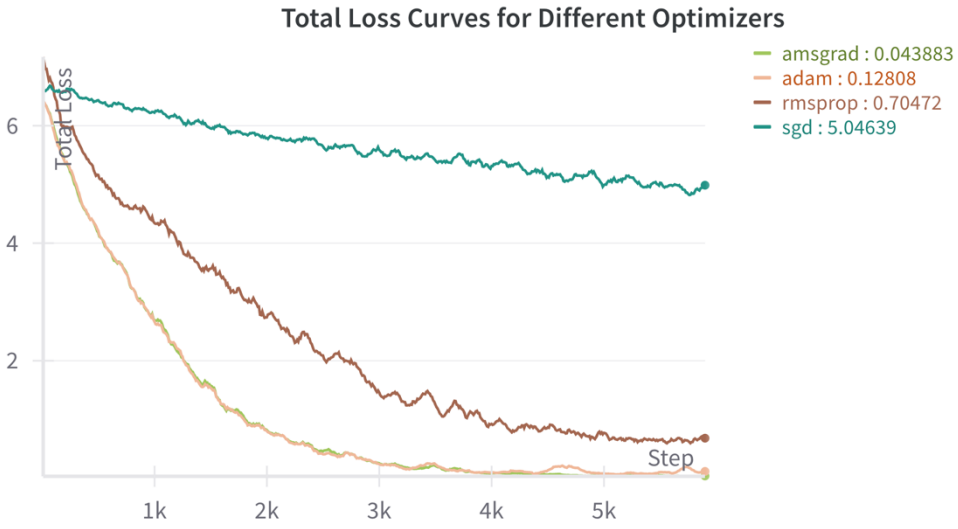


Figure 8: Total loss curves for different optimizers. Note the higher gradient noise with RMSProp and SGD.

Section 4: Summary on overall hyper-parameter tuning, only need to fill up the table below, no more text. (10 marks)

Section	Hyper-parameter	Best configuration (obtained)	mAP	Rank-1
1	Architecture	resnet50_fc512	54.54449	87.00835
2	Data augmentation	Default + Random Perspective	54.60672	87.06794
3.1	Learning rate	0.0001	64.95881	91.06079
3.2	Batch size	64	65.85099	91.95471
3.3	Optimizer	AMSGrad	**65.85099**	**91.95471**

References

[1] F. Schroff, D. Kalenichenko, and J. Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering,” in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 815–823. doi: 10.1109/CVPR.2015.7298682.

[2] A. G. Howard *et al.*, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications,” Apr. 17, 2017, *arXiv*: arXiv:1704.04861. doi: 10.48550/arXiv.1704.04861.

[3] Zakria *et al.*, “Trends in Vehicle Re-identification Past, Present, and Future: A Comprehensive Review,” Feb. 19, 2021, *arXiv*: arXiv:2102.09744. doi: 10.48550/arXiv.2102.09744.

[4] K. Simonyan and A. Zisserman, “Very Deep Convolutional Networks for Large-Scale Image Recognition,” Apr. 10, 2015, *arXiv*: arXiv:1409.1556. doi: 10.48550/arXiv.1409.1556.

[5] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Dec. 10, 2015, *arXiv*: arXiv:1512.03385. doi: 10.48550/arXiv.1512.03385.

[6] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, “Squeeze-and-Excitation Networks,” May 16, 2019, *arXiv*: arXiv:1709.01507. doi: 10.48550/arXiv.1709.01507.

[7] X. Wang, R. Girshick, A. Gupta, and K. He, “Non-local Neural Networks,” Apr. 13, 2018, *arXiv*: arXiv:1711.07971. doi: 10.48550/arXiv.1711.07971.

[8] A. Dosovitskiy *et al.*, “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale,” Jun. 03, 2021, *arXiv*: arXiv:2010.11929. doi: 10.48550/arXiv.2010.11929.

[9] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao, and S. C. H. Hoi, “Deep Learning for Person Re-identification: A Survey and Outlook,” Jan. 06, 2021, *arXiv*: arXiv:2001.04193. doi: 10.48550/arXiv.2001.04193.

[10] Z. Zhong, L. Zheng, G. Kang, S. Li, and Y. Yang, “Random Erasing Data Augmentation,” Nov. 16, 2017, *arXiv*: arXiv:1708.04896. doi: 10.48550/arXiv.1708.04896.

[11] R. Pascanu, T. Mikolov, and Y. Bengio, “On the difficulty of training Recurrent Neural Networks,” Feb. 16, 2013, *arXiv*: arXiv:1211.5063. doi: 10.48550/arXiv.1211.5063.

[12] A. Devarakonda, M. Naumov, and M. Garland, “AdaBatch: Adaptive Batch Sizes for Training Deep Neural Networks,” Feb. 14, 2018, *arXiv*: arXiv:1712.02029. doi: 10.48550/arXiv.1712.02029.

[13] S. Ruder, “An overview of gradient descent optimization algorithms,” Jun. 15, 2017, *arXiv*: arXiv:1609.04747. doi: 10.48550/arXiv.1609.04747.

[14] S. J. Reddi, S. Kale, and S. Kumar, “On the Convergence of Adam and Beyond,” Apr. 19, 2019, *arXiv*: arXiv:1904.09237. doi: 10.48550/arXiv.1904.09237.

Rubric

Section 1: Familiarity with the code provided (30 marks)

Task 1.1: Running and understanding the code (10 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.

(3 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.

(2 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 (25 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 (5 marks)

- **Combining techniques and experimentation**
  - *Criteria*: Correct implementation of combined augmentation techniques.
  - *Indicators*: Code successfully combines the techniques and executes without errors.

(3 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.

(2 marks)

Section 3: Exploration of hyperparameters (25 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)

Task 3.2: Optimizer exploration (5 marks)

- **Experimentation with different optimizers**
  - *Criteria*: Correct experimentation with an additional optimizer.
  - *Indicators*: Clear documentation of the chosen optimizer and implementation using the best-performing LR and BS.

(2 marks)

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

(3 marks)

Section 4: Completing the summary Table (10 marks)

- **Consistency with Experimental Results**
  - *Criteria:* The hyperparameters listed must align with findings from previous sections. Each for **2 marks**.
  - *Indicators:* No contradictions between reported values and earlier discussion.

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