Brute Forcing Keypoints: BoVW vs CNN

Jonathon Dilworth
School of Computer Science
University of Manchester, UK
jonathon.dilworth@postgrad.manchester.ac.uk

***Abstract*—Recent advancements in deep learning enable neural architectures that can automatically extract features as a natural by-product of their execution. For image classification, this is most notably apparent in convolutional neural networks, which have seen significant attention as GPU architecture has matured. However, the focus on neural representation learning has diverted attention away from classical techniques. This work poses the question: to what extent can modern GPU hardware paired with software libraries such as cuPy and cuML improve classical CV methods? We revisit Bag-of-Visual-Words (BoVW) for image classification, using cuML to scale codebook construction to 50 million keypoints on CIFAR-10. Our best BoVW configuration matches a modernised LeNet-5 variant for classification accuracy, but falls short of a more powerful, compute-intensive, VGG-16 with batch normalisation. Ultimately, our results signal that while modern hardware enables previously impractical scaling for classical methods, the fundamental limitations of BoVW (particularly vector quantisation error and the absence of spatial hierarchy) remain when compared to deeper architectures. Future work could explore more sophisticated classical approaches. Source code and reproducible artefacts are made available at https://github.com/jonathondilworth/uom-vision.**

***Index Terms*—Bag of Visual Words, Image Classification, CNN, GPU Acceleration, SIFT, cuML**

I. Introduction

For the last decade, advancements in Computer Vision (CV) have focused on deep learning-based approaches. However, in this work, we revisit an earlier method in image classification that predates AlexNet [1], Bag of Visual Words (BoVW) [2]. We benchmark our GPU-based implementation (that leverages up to 50,000,000 keypoints in codebook construction, enabled by H100 GPU architecture) against a 'modernised' LeNet-5 [3] and a VGG-16 CNN [4]. We use two well-known datasets for training, tuning and evaluation: CIFAR-10 and CIFAR-100 [5]. This comparative study reviews several architectural components of both approaches and explores hyperparameter tuning and adaptive learning rates.

II. Related Work

A. Traditional Computer Vision

Csurka et al. [2] drew from information retrieval by framing image classification in terms of a Bag-of-Visual-Words (BoVW). Their approach applies k-means to local features detected with Harris-Affine [6] and SIFT [7], thereby constructing a visual lexicon; pooled keypoints then form histograms and are fed to an SVM. The key insight was to represent unordered keypoint counts–occurrences of similar local features–as a single vector-space representation. Their method demonstrated a macro-average accuracy of $\approx 96\%$ on Caltech-4/5, outperforming the then state-of-the-art 93% reported by Fergus et al. [8]. However, their approach has at least two significant limitations. First, the error introduced through vector quantisation becomes increasingly apparent as the number of classes grows. Second, the BoVW fails to capture any spatial context between image keypoints. These shortcomings inspired spatial pyramid matching [9] and, later, hierarchy-based learned representations via deep convolutional neural networks [1].

B. Convolutional Neural Networks

Image classification using convolutional neural networks (CNNs) is not a novel approach in computer vision; it predates techniques such as BoVW by over a decade. LeCun et al. [10] presented a CNN for classifying handwritten ZIP codes with a test error of $\approx 1\%$ (with $\approx 9\%$ reject rate)—*considered state-of-the-art at the time despite small-scale data and CPU-bound compute limitations.* The architecture would continue to be adapted for MNIST and became known as LeNet-5 (Fig. 1) in LeCun's paper on gradient-based learning [11], [3], which provided the necessary foundations for GPU-driven breakthroughs with deep networks over a decade later.

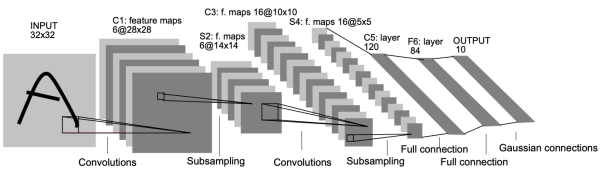


Figure 1: The Convolutional Neural Network architecture from LeCun et al., [3] used for classifying hand written and machine-typeset characters.

AlexNet [1] achieved a 15.3% top-5 error in the ImageNet ILSVRC-2012 contest—a 10.9-point gain over the runner-up. Importantly, their network architecture enabled model training to be efficiently parallelised over two GPUs, allowing for increased network depth and many more trainable parameters. Introducing techniques like Rectified Linear Units (ReLU) [12], dropout regularisation [13], and max-pooling helped reduce model degradation and overfitting. These key insights sparked a surge in deep-net research and GPU-accelerated computing, motivating later work such as VGG-16/19 [4] and the residual-learning breakthrough, ResNet [14].

III. Methods

A. Experimental Design

Experiments are repeated three times, and the average measure is taken to yield our experimental results. Visualisations are based on averages but may also indicate variance where appropriate. A random seed is used to ensure reproducibility and is drawn from $seeds = (42, 84, 168)$ for each experiment.

Since CIFAR10 and CIFAR100 are assembled from uniformly distributed classes of equal importance, we deem accuracy and macro F_1 as appropriate performance metrics [15]. We verify that measured accuracies equate *(approximate to within* 0.5%*)* to micro F_1 to help protect against inadvertently introducing class skew during data augmentation. An additional check is made to ensure accuracy measures do not exceed macro

F_1 as a *sanity check* when extracting and reshaping our experimental data from result logs. Notions of time and space complexity are *informally* considered in evaluating our proposed approaches to both hyperparameter tuning and image classification tasks.

CUDA was used in experiments for both classical and neural network-based methods. We used cuML and cuPy for BoVW-based clustering and classification, and PyTorch for our CNN implementations. The workload was distributed across two machines, allowing for several thousand runs *($\approx$ 2900)* for BoVW tuning and the use of sweeps in CNN-based approaches. The codebase, details related to machine specification and a complete account of the logs are provided under Appendix.A.

B. Data & Preprocessing

CIFAR10 and CIFAR100 [5] are selected due to their maturity in the research area, allowing us to position our results within the context of the State-of-the-Art (SoTA) for either task *(multiclass classification in images, CNN-based VS traditional CV)*. Additionally, we assume image classification on CIFAR10 is challenging enough to effectively compare BoVW to a relatively simple CNN and hypothesise that performance on CIFAR100 will demonstrate relative strengths of selecting a CNN-based approach, allowing for a meaningful discussion in Section.VI. CIFAR10 is used for tuning, and both datasets are subsequently trained and evaluated. Details of each dataset are outlined under Table I.

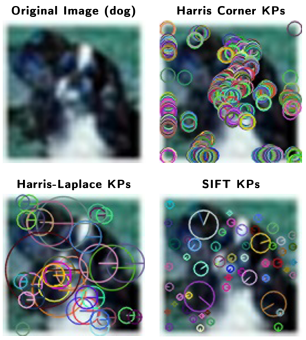
Property	**CIFAR-10**	**CIFAR-100**
Image Dimensions	$3 \times 32 \times 32$	$3 \times 32 \times 32$
Number of classes	10	100 (20 coarse)
Label granularity	Fine (10)	Fine (100) & Coarse (20)
Training Set Size	50000 *(5k per class)*	50000 *(500 per class)*
Test Set Size	10000 *(1k per class)*	10000 *(100 per class)*
Applied Splits		
Training	35000	35000
Validation	15000	15000
Test (hold-out)	10000	10000

Table I: Data statistics: CIFAR10 and CIFAR100 image–classification datasets.

We perform a 70:30 training-validation split across CIFAR10's training set, which is used for hyperparameter tuning and training. The test set is kept as a hold-out set that is not evaluated until models are appropriately tuned and experimental results are drawn.

Preprocessing techniques vary depending on the experiment, the models used, and the algorithms employed by each approach. For example, images may or may not be converted to grayscale, scaled or transformed, depending on (1) the downstream algorithm responsible for processing them, (2) the architectural requirements (VGG-16 necessitates upscaling), and (3) the optimal approach, dictated through hyperparameter optimisation and our ablation study (Section. IV).

In constructing two approaches for comparison, BoVW and CNN-based, several methodological considerations must be made. We outline our methods in subsequent subsections and provide a short discussion justifying our decisions.



(a)

Detector	**Keypoints**
Harris	424
Harris–Laplace	36
SIFT	64

(b)

Figure 2: Properties of various keypoint detectors; (a) visualisation of detected keypoints using techniques: Harris Corner Detector, Harris–Laplace, and SIFT; (b) the number of key-points returned by each detector.

C. Bag of Visual Words Pipeline

1) Detectors & Descriptors: We expose APIs for the Harris corner detector (referred to as 'Harris' within plots) [16], Harris-Laplace Detector [6], and OpenCV's native SIFT [7] implementation within our codebase (and include an external library for Dense SIFT [17]). This allows us to sweep multiple potential configurations during tuning and ablation (discussed under Section. IV).

We include these implementations *(excluding dense SIFT due to architectural incompatibilities with our implementation)* in our analysis to demonstrate their associated strengths and weaknesses. For instance, both SIFT and Harris-Laplace are invariant to scale, and SIFT is partially robust to some affine transformations [7]. Conversely, the Harris corner detector is weak to both. We expect the corner detector-SIFT combination will saturate the number of keypoints (Figure. 2 shows the number of keypoints produced relative to each method), resulting in either a noisier histogram representation prior to classification. Thus, we hypothesise that using the Harris corner detector will lead to a performance penalty, and SIFT *(for detection and description)* is most likely to perform well due to its overall robustness and quality of keypoint detection.

2) Clustering Algorithm: While k-means is typically thought of as the de facto choice of clustering algorithm in BoVW pipelines [2], alternative approaches to codebook construction include variations on Gaussian-Mixture Models (GMMs) [18], Decision Trees (DT) [19] and alternative hierarchical clustering algorithms.

Within our work, k-means is implemented as a baseline clustering algorithm, as it provides a simple means of ensuring our implementation functions as intended. Moreover, k-means is computationally efficient when compared to GMMs, which rely on Expectation-Maximisation (EM) to converge. However, GMM's ability to provide soft assignments is a potential advantage in cases where class labels may not be so granular and share similar visual properties (e.g. CIFAR100, coarse and fine labels) [18]. DT-based clustering methods, specifically ERC-Forests [20], seem promising. They offer better computational efficiency and highly discriminative codebooks, minimising the loss during vector quantisation [21] and can contribute to improved downstream classification [20].

We note these alternative quantisation methods to illustrate different approaches' relative strengths and weaknesses. Ideally, we would implement each quantisation method; ERC-

Forests for use on CIFAR10, and GMMs on CIFAR100. Unfortunately, only an implementation of *k*-means is provided due to time constraints.

3) Histogram Encodings: can take the form of raw counts, though not typically considered for use with downstream SVM classification to avoid performance penalties associated with high variance in the feature space. Thus, we explore several potential histogram encoding and normalisation strategies, detailed below.

a) L2 Normalisation: provides one means of preparing our histogram representations prior to downstream classification, since we know our bins satisfy $\mathbb{R}_{\geq 0}$. Moreover, the L_2 does not implicitly perform feature selection (in contrast to L_1), which may be important to protect against overfitting *(in the case of non-linear kernels)*.

b) Term frequency-Inverse document frequency: (TF-IDF) [15] may be advantageous as it strengthens features most distinctive across the dataset. Through applying this information retrieval-based method, *see equations:* (1a), (1b), (1c), we might strengthen *visual words* with high class discriminability, while dampening regular occurrences often seen across all samples.

$$tf(t,d) = \frac{f_d(t)}{\sum_{t \in d} f_t(d)} \tag{1a}$$

$$idf(t,D) = \log\left(\frac{|D|}{d \in D : t \in d}\right) \tag{1b}$$

$$tfidf(t,f,D) = tf(t,d) \cdot idf(t,D) \tag{1c}$$

c) Hellinger Approximation: Assuming we retain the raw counts for any given histogram, *required for* (1a) $\rightarrow$ (1c), we can approximate the Hellinger kernel [22] (2a) efficiently [22] through applying (2b) to our histogram representations (prior to SVM classification). Interestingly, all three encoding techniques are composable (3). Thus, we apply each histogram encoding in turn and provide our findings under Section. 2.

$$k(x,y) = \sum_i \sqrt{x_i} \cdot \sqrt{y_i} \tag{2a}$$

$$hellinger : x \rightarrow \frac{\sqrt{x}}{\|\sqrt{x}\|_2} \tag{2b}$$

$$(tfidf \leftarrow hellinger) \Rightarrow \frac{\sqrt{tf \cdot idf}}{\|\sqrt{tf \cdot idf}\|_2} \equiv L_2\left(\sqrt{tf \cdot idf}\ \right) \tag{3}$$

4) Classifiers: SVM-based classification pipelines typically perform well [2] due to their discriminative properties—*explicitly maximising the margin between decision boundaries*. However, the linear SVM may be insufficient to capture separability in the feature space after vector quantisation, i.e, smaller codebooks may lead to underfitting, whereas larger codebooks ought to yield higher classification accuracy *(assuming sufficient data and appropriate regularisation)*. Given [22], we believe combining the Hellinger approximation (2b) with a linear SVM ought to yield comparable performance to non-linear kernels *(assuming ample complexity in the feature space)*. We decide to investigate both linear and Radial Basis Function (RBF) SVMs [23], since it would be interesting to compare the two and to validate or invalidate our beliefs. Additionally, both have the added benefit of cuML support, enabling training via GPU acceleration.

D. Neural Network Architectures: CNNs

For our networks, we decide to implement two distinct architectures, a LeNet-5 [3] variation ('M-LeNet-5-D', or 'Modern LeNet-5 with Dropout'), and a VGG-16 [4] model implementing batch normalisation [24] ('VGG-16-BN').

In the remainder of this subsection, we outline (1) our chosen CNN architectures, (2) the optimisation and regularisation methods employed, and (3) the hyperparameter exploration approach.

1) M-LeNet-5-D: First, LeNet-5 is an early (shallow) CNN implementation (See Figure. 1), making it a good candidate for establishing baseline performance. The topology is designed to accept images with a resolution of 32×32 [10], making it convenient for use with CIFAR-10 and CIFAR-100. Additionally, it is conceptually simple, can be implemented with relative ease and trained in minutes on modern hardware, allowing us to verify our workflow (necessary for implementing sweeps and tuning hyperparameters via wandb). Thus, it is an appropriate choice of architecture prior to implementing networks of increasing depth that may take several hours to train.

We adapt LeNet-5 [3] to accept RGB images, since we intend to process real-world scenes and colour may carry crucial semantic cues. Additionally, we opt to employ the ReLU [12] activation function *(rather than weighted tanh)* to aid in improving convergence speed and avoid vanishing gradients. Dropout [13] is implemented at the fully connected layers, and weight decay is utilised to avoid overfitting. Max pooling is preferable to average pooling [1] and is applied during *downsampling* to capture more distinctive features. We opt out of using an antiquated RBF-based output layer and instead adopt softmax for classification, applying cross-entropy loss for training [25]. Finally, we employ momentum for fixed optimisation strategies [26], though we sweep over schedulers using *wandb* [27] (Section IV-B) to investigate whether adaptive learning rate, momentum and alternative optimisation strategies [28] may offset some limitations of network depth and breadth. An overview of our network architecture, its features, fixed parameters and sweep space is presented in the Table. II.

Network Architecture: M-LeNet-5-D	
Input	$3 \times 32 \times 32$ RGB image
Conv1	6 @ 5×5, ReLU
Pool1	MaxPool 2×2
Conv2	16 @ 5×5, ReLU
Pool2	MaxPool 2×2
FC1	120 units, ReLU, Dropout ($p = 0.3$)
FC2	84 units, ReLU, Dropout ($p = 0.3$)
FC3 (output)	10 units, Softmax
Fixed Training Settings	
Optimiser	SGD with momentum ($\mu = 0.9$)
Initial LR	0.01
Loss	Cross-entropy
Weight decay	L_2 ($\lambda = 5 \times 10^{-4}$)
Pooling	Max pooling
Activation	ReLU
Hyperparameter Exploration	
Search Strategy	Random
Learning rate	$\{[0.001, 0.01], (0.00, 0.01), 0.01, 0.001, 0.0001\}$
Momentum	$\{[0.80, 0.95], 0.80, 0.85, 0.90, 0.95\}$
Scheduler type	{StepLR, CyclicLR, CosineAnnealingLR}

Table II: M-LeNet-5-D Architecture, Training and Hyperparameter Space

2) VGG-16-BN: The second proposed architecture is a standard VGG-16 [4] model with batch normalisation (VGG-16-BN). VGG-16 was originally published citing the effect of increased network depth on classification performance [4]. Batch normalisation variants were proposed later in [24] (as implemented in [29]). We deem VGG-16-BN an appropriate model to include within our study due to the distinctive difference in network depth compared to M-LeNet-5-D. Though, we note that as network depth increases, the architecture typically benefits from the introduction of residual blocks *(skip layers)*. Without the inclusion of such techniques, we may be placing a ceiling on the models' performance. An overview of the network architecture, training strategies and hyperparameter search space is provided in Table. III.

Network Architecture: VGG-16-BN	
Input image	$3 \times 227 \times 227, RGB$
Conv Block 1	$2 \times (64, (3,3), BN, ReLU) \rightarrow MaxPool(2,2)$
Conv Block 2	$2 \times (128, (3,3), BN, ReLU) \rightarrow MaxPool(2,2)$
Conv Block 3	$3 \times (256, (3,3), BN, ReLU) \rightarrow MaxPool(2,2)$
Conv Block 4	$3 \times (512, (3,3), BN, ReLU) \rightarrow MaxPool(2,2)$
Conv Block 5	$3 \times (512, (3,3), BN, ReLU) \rightarrow MaxPool(2,2)$
Fully Connected 1	$(4096, ReLU, Dropout(p = 0.5))$
Fully Connected 2	$(4096, ReLU, Dropout(p = 0.5))$
Final Layer (FC3)	$(10, Softmax)$
Fixed Training Details	
Loss function	Cross-entropy
Batch normalization	After every convolution
Activation	ReLU
Pooling	Max pooling
Hyperparameter Sweep Space	
Search Strategy	Random
Learning rate	{0.05, 0.025, 0.01}
Momentum	{0.85, 0.90, 0.95}
Weight decay (L_2)	{0.0004, 0.0005, 0.0006}
Optimiser	{SGD, Adam, RMSprop}
Step size	{25, 30, 35}

Table III: VGG-16-BN Architecture, Training Details, and Sweep Space

IV. Hyperparameter Tuning & Ablation Study

This section discusses strategies for hyperparameter tuning and architectural component selection (ablation study). We note that carefully choosing appropriate hyperparameters is essential to achieving sufficient model performance. In some cases, trade-offs between computational efficiency and optimal performance must be made.

For BoVW [2], we first tune *k*-means' K value [25] *(or vocabulary size)*, compare histogram encoding strategies and include a systematic grid search over C *(linear SVM)* and γ for an RBF SVM classifier [23], plotting the results against the validation accuracy. We apply a similar initial step for our CNN architectures' learning rate and momentum. Additionally, we analyse the effects of various architectural changes: detector-descriptor combinations, scaling and augmentation in BoVW; and optimisation & scheduling strategies: SGD [26] [28], RMSProp [30], ADAM [31], AdaGrad [32], Step-LR [33], Cyclic-LR [34], and Cosine-Annealing [35] in the CNN architectures, measuring their effects on validation accuracy and producing visualisations. In the case of CNNs, we opt to utilise the 'weights and biases' (wandb) sweeps API [27].

It is important to note several fundamental assumptions made before performing these grid searches, sweeps and architectural analyses. We assume:

1) Measures of validation accuracy taken at medium resolution scaling $(32, 32) \rightarrow (64, 64)$ with applied linear or cubic interpolation, at a relatively low vocabulary size, are likely representative of measures taken as those parameters are scaled up. This assumption is made (in part) due to the practical limitations of computational efficiency and time.
2) Performing a random walk over the search space (for CNNs) will more efficiently approximate a systematic grid search, as discussed in [36].

Additionally, our approach to tuning CNN-based models is comprised of a two-step method: (1) perform a random walk through a relatively high dimensional configuration space (i.e. standard optimisation hyperparameters, optimisation parameters and scheduler parameters), then (2) select top-5 candidate solutions based on prior sweeps and apply subsequent tuning. The following subsections provide further details of the above-mentioned approaches, visualisations, tuning results and brief analyses.

A. Bag of Visual Words

1) Vocabulary Size (k-means Clustering): First, we evaluate the effects of increasing *k*-means' K parameter (vocabulary, or *codebook* size) on validation accuracy, using a Linear and RBF SVM, with $C = 1.0, \gamma = 0.1$ (assumed to be a reasonable regularisation penalty and margin size).

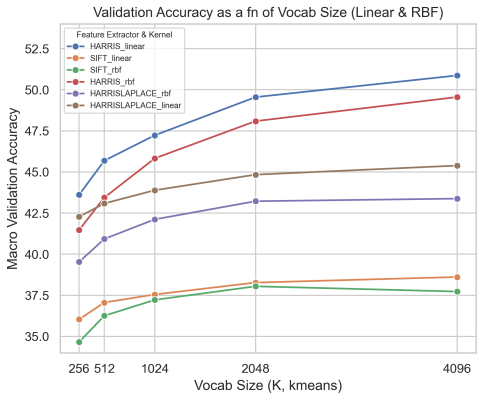


Figure 3: Effect of codebook vocabulary size on the validation accuracy for classifier-keypoint detector combinations

Our expectation (hypothesis) was verified *(for the most part)*: accuracy *(generally)* increases as K increases (Figure. 3), eventually plateuing and decreasing once K exceeds a certain threshold ($K \simeq 2048$).

However, we did not expect SIFT's performance to be so poor compared to other detector-descriptor variants. We note that these variants produce more keypoints at the cost of scale sensitivity (Harris corner detector) and particular weakness to affine transformations (Harris-Laplace). Thus, we might suppose that producing more keypoints for image recognition where class instances are represented at a *(broadly)* similar scale with limited variance in skew is preferable [37].

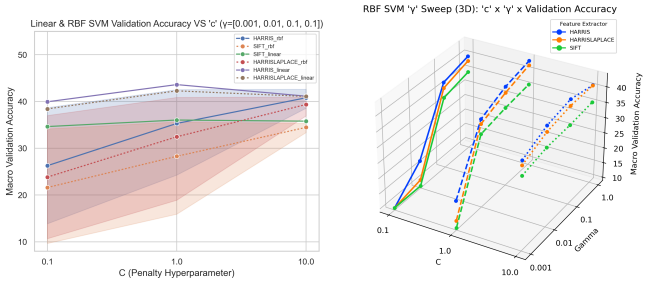
2) Histogram Encodings: In addition to vocabulary size, we review the effects of modifying the histogram encoding scheme when employing an RBF-SVM and the Harris-SIFT detector-descriptor combination. We demonstrate that employing Hellinger-L2 [22] [38] marginally outperforms all other encoding schemes, yielding fairly competitive accuracy scores

when selecting an appropriate value for K *(codebook size)* and applying image scaling $(32, 32) \rightarrow (128, 128)$.

Seed	L2	TFIDF-L2	Hellinger-L2	TFIDF-Hellinger-L2
42	53.52	53.48	55.28	55.18
84	53.87	53.98	55.56	55.45
168	53.80	53.75	55.20	55.28
Average	**53.73**	**53.75**	**55.35**	**55.30**

Table IV: Validation accuracy (%) for different histogram encodings and normalisation mechanisms.

3) SVM Classifers (Linear & RBF): The tuning step for the BoVW image classifier involves performing a grid search over $C = (0.1, 1, 10)$ for both linear and RBF SVMs, and $\gamma = (0.001, 0.01, 0.1, 1)$ for the RBF SVM. For purposes of computational efficiency, we set k-means' $K = 256$ and apply image scaling $(32, 32) \rightarrow (64, 64)$. We include both Linear and RBF results on Figure. 4a, providing additional intuition for the RBF-SVM *(sweeping over gamma)* in Figure. 4b.



(a) 2D **(b)** 3D

Figure 4: Sensitivity analysis: performing a grid-search sweep over both Linear & RBF SVMs with $C = [0.01, 0.1, 1.0]$ & $\gamma = [0.001, 0.01, 0.1, 1.0]$.

4) Augmentation & Scaling: In brief, whilst we were careful to increase each class distribution uniformly during our data augmentation procedure, we found that this did not effectively modify the performance of the BoVW method (Figure. IV). However, we note this is likely due to the selected augmentation techniques (having only implemented random flip and brightness transformations).

While the effects of scaling the input images (Figure. IV) essentially confirm what we already know, i.e. both Harris-Laplace & SIFT are scale invariant *(and Harris corner detector is not)*, there is a fairly obvious caveat, we see a substancial gain in performance upscaling from $(32, 32) \rightarrow (64, 64)$ but no significant benefit thereafter, except in the case of the corner detector that demonstrates some fairly significant gain, up until a point ($p_{scale} = (128, 128)$).

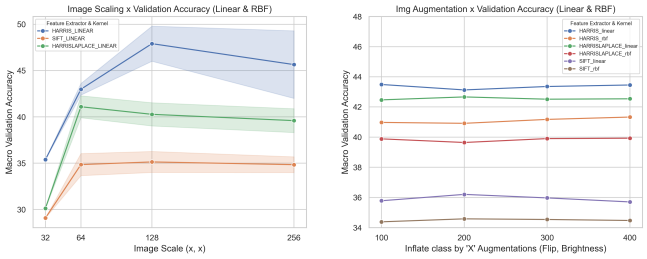


Figure 5: Results from tuning scaling and data augmentation components.

B. Convolutional Neural Networks

1) Tuning M-LeNet-5-D: First, we conduct a random walk through the space (described in Table. II) [36] using a wandb sweep [27]. Having produced candidate solutions, we rank them by validation accuracy and note the Mode configuration across the top five candidates (Table. V. Subsequent steps involve conducting a grid search to tune the selected candidate configuration.

Top-5 Candidate Configurations M-LeNet-5-D

ID	Schedule	LR_{range}	Mo_{μ}	D_o	V.Acc
78	cyclic	$[0.001, 0.01]$	$[0.8, 0.95]$	Yes	64.55
75	cyclic	$[0.001, 0.01]$	$[0.8, 0.95]$	Yes	64.00
67	cosA_LR	$(0.00, 0.01)$	0.80	Yes	63.28
71	cyclic	$[0.001, 0.01]$	$[0.8, 0.95]$	Yes	62.03
79	steplr	0.01	0.80	Yes	61.94
AVG	-	-	-	-	**63.16**
M	cyclic	$[0.001, 0.01]$	$[0.8, 0.95]$	Yes	-

Table V: Configuration and performance of Top-5 candidates from an initial M-LeNet-5-D wandb sweep. LR_{range}: learning rate range from lower bound (*base_lr*) to an upper bound (*max_lr*); the same is applied for momentum: Mo_{μ}. D_o indicates use of dropout; V.Acc measures are provided.

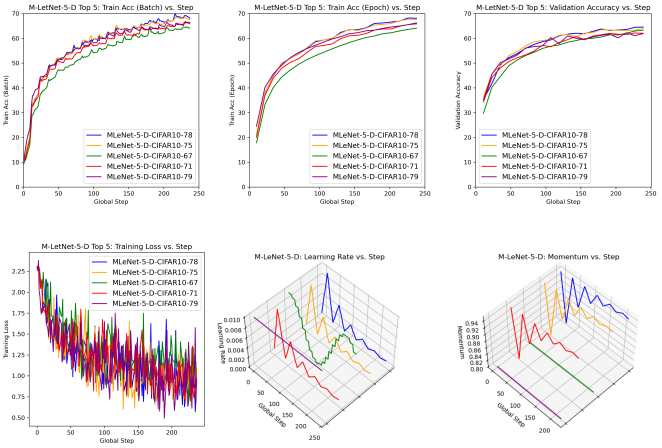


Figure 6: A multiplot presenting the training & validation accuracy, training loss and visualisation of learning rate & momentum for the top five candidates from Table. V.

We observe that the Cylic Learning Rate (CLR) [39] configuration is the most highly ranked, outperforming Cosine Annealing [35] and StepLR [28]. Additionally, by observing Figure. 6, we note a distinctive feature: **momentum acting inversely to learning rate** [39]. This observation provides good intuition, since setting a high (constant) momentum with a high (constant) learning rate (and vice versa) is not typically desirable. We assume this also offsets the possibility of exploding gradients (resulting from an out-of-control learning rate). We refer to [39] and review related publications [40], verifying our initial intuition and providing further insights.

Further analysis of Figure.6 shows validation accuracy closely tracking training accuracy. Thus, the model may not have converged or may be underfitting. Each model might benefit from resumed training. We note that the learning rate and momentum have already decayed in the case of CLR (as our initial sweep configuration failed to span all cyclic modes [39]). Thus, to tune CLR we (1) retrain, increasing the number of epochs by a factor of 10, (2) modify the range of decay step values to (2, 20, 200), and (3) use alternative cycle modes. Our measures are presented in Figure. 7.

Our initial intuition would have suggested that if training were to be resumed, the decay rate for the scheduler would

need to be adjusted. We initially thought that failure to adjust under the *'triangular2'* CLR mode (where an annealing effect can be observed) would direct the model towards premature convergence. However, while that may be true if an inappropriate lower and upper bounds are selected (learning rate & momentum; i.e. forcing the model to a suboptimal space and relying on the base configuration to continue working), upon reflection (and further reading [41]), we consider model convergence in a single cycle (i.e., *'super-convergence'* [41]) as a potentially desirable feature.

To illustrate the central thesis in [41], we reference the additional tuning runs for CLR (Figure 7). The highest validation accuracy is obtained fairly early by *TRI-STEP-20*, the mode in which no decay occurs. Thus, our experiments seem to validate the effectiveness of *'super-convergence'*, i.e. if model performance is discovered early, quenching exploration is advisable to avoid model degeneration. Finally, the training accuracy continues to increase, which correlates to the slow decrease in validation accuracy. Eventually, the validation accuracy reaches the same position as our baseline (SGD with no scheduler). Thus, we assume the network topology (shallow depth and breadth) is now the fundamentally limiting factor. Further discussions of our results are provided in V.

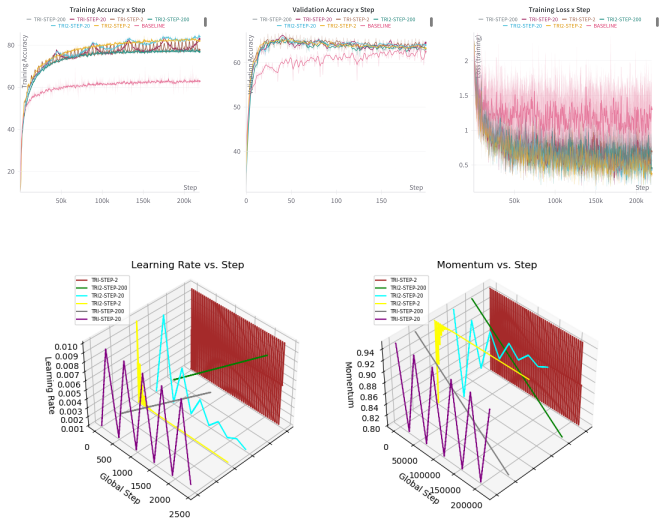


Figure 7: A multiplot presenting the training & validation accuracy, training loss and visualisation of learning rate & momentum for values of CyclicLR, cyc_mod = TRI ∨ TRI2 & step_size_up = [2, 20, 200], V.

2) Tuning VGG-16-BN: We recognise that time constraints (primarily) force us to constrict the search space, as shown through Table III. Nevertheless, we ran a sweep for approximately twelve hours (Fig. 8 and obtained interim findings for model configuration (Table. VI). The best run achieved a validation accuracy of ≈ 83%, which we consider reasonably competitive and a strong foundation for future refinements.

ID	LR	Mo$_\mu$	Decay$_w$	Optimiser	Schedule	Step	V. Acc
100	0.01	0.90	0.0004	RMSprop	StepLR	30	82.89
99	0.025	0.85	0.0004	SGD	StepLR	35	81.36
101	0.01	0.90	0.0006	Adam	StepLR	25	80.26
97	0.01	0.95	0.0006	RMSprop	StepLR	30	75.63
98	0.05	0.85	0.0006	Adam	StepLR	25	67.03
AVG	-	-	-	-	-	-	**76.62**

Table VI: Top 5 VGG-16-BN Sweep Configurations on CIFAR-10

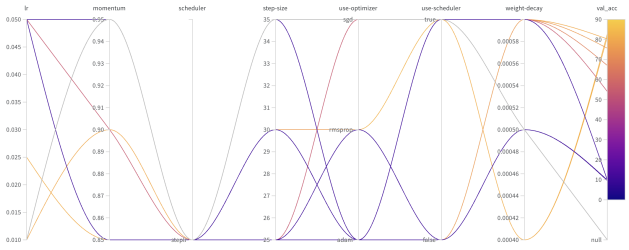


Figure 8: A parallel coordinates plot generated using *wandb sweeps* for *VGG-16-BN*.

V. Experimental Results

A. CIFAR10

Table VII summarises our results for CIFAR-10. The BoVW configuration with the highest accuracy measure *(Harris–SIFT, Hellinger-L_2, $k = 4096$, linear SVM, $C = 1$)* achieves 65.46% accuracy on the hold-out set. Broadly speaking, the use of an RBF kernel *(in our specific context)* yields no effective performance gain, as our RBF measures show 64.83% on the same partition (a pattern that can be seen throughout). M-LeNet-5-D reaches 66.31% validation accuracy after only forty epochs, yielding 64.58% on the hold-out set, effectively matching the performance of our best BoVW implementation, despite its relatively tiny set of trainable parameters. Moreover, M-LeNet-5-D is **much more** computationally efficient in both time and space when compared with our best BoVW pipeline. The narrative changes, however, for our VGG-16-BN model which is computationally demanding, but delivers an increase in accuracy for both the validation and test set, measured at 87.56% and 83.90%, respectively. This gulf in accuracy effectively highlights the advantage of increased network depth and hierarchical representation learning compared with BoVW.

CIFAR10		
Configuration	**Val. Acc**	**Test. Acc**
BoVW (Linear SVM)		
$C = 1$, $H_k = 0.2$, $k = 2048$	57.87	59.17
$C = 1$, $H_k = 0.18$, $k = 2048$	59.70	60.62
$C = 1$, $H_k = 0.04$, $k = 4096$	64.33	65.46
BoVW (RBF SVM)		
$C = 1, \gamma = 1$, $H_k = 0.2$, $k = 2048$	57.86	59.19
$C = 1, \gamma = 1$, $H_k = 0.18$, $k = 2048$	59.67	60.60
$C = 1, \gamma = 1$, $H_k = 0.04$, $k = 4096$	64.32	64.83
CNN (MLeNet-5-D)		
40 epochs, cyclicLR $ID = 171$	66.31	-
200 epochs	64.69	64.58
CNN (VGG-16-BN)		
10 epochs, stepLR, $ID = 100$	82.89	83.90
200 epochs	–	–

Table VII: Results on CIFAR-10: BoVW (SVM) and CNN Architectures; H_k denotes the Harris k free parameter

B. CIFAR100

The BoVW-based implementation failed to evaluate on CIFAR-100 (Table VIII) due to memory constraints *(discussed in Section. VI)*. M-LeNet-5-D struggles with CIFAR100, likely due to its inability to capture rich intermediate or global features (due to its shallow depth) associated with a significantly larger number of classes. Thus, its performance ceiling sits at ≈ 30%, achieving a final test accuracy of 28.71% after 200 epochs, though the model converged prior to terminating. VGG-16-BN provides evidence that deeper architectures scale more effectively (at the cost of compute), whose accuracy we

measure at 59.78% after 10 epochs. We opt not to repeatedly train over many epochs due to the associated financial cost of cloud compute, but would expect marginal performance gain over 59.78% prior to introducing residual skip layers, improving regularisation, increasing depth and finely tuning optimisation strategies and schedulers.

CIFAR100		
BoVW (Linear & RBF SVM)		
All simulations exhausted available memory for both the RTX 4000 ADA & the H100 cloud instance		
Configuration	**Val. Acc**	**Test. Acc**
CNN (MLeNet-5-D)		
10 epochs, cyclicLR, $ID = 133$	17.05	16.72
200 epochs, cyclicLR, $ID = 134$	28.51	28.71
CNN (VGG-16-BN)		
10 epochs, stepLR, $ID = 3$	60.28	59.78
200 epochs	–	–

Table VIII: Results on CIFAR-100: BoVW (SVM) and CNN Architectures

VI. Discussion

Broadly speaking, our results confirm known trends in image classification, i.e. deeper CNNs typically outperform shallower ones and BoVW-based pipelines on complex datasets. Nonetheless, we draw some insights below.

While the accuracy of our BoVW pipeline is relatively modest compared to (traditional-based) SoTA CV methods (e.g. discriminative pooling [42]), it may be considered reasonably competitive, achieving similar scores to methods such as dual codebooks [43]. However, this approach exploits the inherent trade-off between accuracy and efficiency. By combining Harris corner detector with SIFT descriptors, we yield up to 50,000,000 keypoints on CIFAR10 before codebook construction, essentially brute forcing accuracy scores. This is made possible through CUDA-based implementations of traditional ML algorithms and best-in-class hardware *(i.e. the use of H100 GPU architecture)*. Thus, we do not deem it a practical solution.

Having experimented with variations on standard architectures (M-LeNet-5-D and VGG-16-BN) in our CNN-based methods, we assume the shallower network is fundamentally limited by its topology (depth and breadth). While the network appears to be learning adequate local *(and perhaps intermediate)* representations, yielding commendable performance on CIFAR10, we do not believe it has enough trainable parameters or pooling depth to acquire global spatial context. As such, the performance on CIFAR100 is far from desirable. Whereas the performance increases significantly (as expected) for VGG-16-BN, though the failure to utilise advanced methods *(such as skip connections)* likely imposes a ceiling on the model performance. This is especially noticeable on CIFAR-100, where deeper architectures with residual blocks would likely be more appropriate.

VII. Conclusions & Further Work

We find that generating large numbers of keypoints allows for effective, though impractical, boosts to classification accuracy in BoVW pipelines (on relatively small datasets with a limited number of classes). The application of the Hellinger approximation [22] prior to classification yields better performance scores in some cases on BoVW. Our CNN-based results are consistent with contemporary academic CV literature. Finally, we reaffirm the position in [41]; i.e., using effective scheduling in adaptive learning rates for CNN-based architectures leads to faster convergence.

In the future, we plan to investigate more advanced network architectures in detail. We also intend to continue working on our implementation (framework) for running CV workflows, incorporating modules that enable transfer learning and more advanced methods for BoVW. Finally, we look to investigate strategies for highly parallelised, distributed GPU workloads.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. J. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25. Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

[2] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," *Work Stat Learn Comput Vision, ECCV*, vol. Vol. 1, Jan. 2004.

[3] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/726791

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] A. Krizhevsky, G. Hinton, and others, "Learning multiple layers of features from tiny images," 2009, publisher: Toronto, ON, Canada.

[6] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *International journal of computer vision*, vol. 60, pp. 63–86, 2004, publisher: Springer.

[7] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004, publisher: Springer.

[8] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, vol. 2. Madison, WI, USA: IEEE Comput. Soc, 2003, pp. II–264–II–271. [Online]. Available: http://ieeexplore.ieee.org/document/1211479/

[9] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, vol. 2. New York, NY, USA: IEEE, 2006, pp. 2169–2178. [Online]. Available: http://ieeexplore.ieee.org/document/1641019/

[10] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989. [Online]. Available: https://direct.mit.edu/neco/article/1/4/541-551/5515

[11] L. Bottou, C. Cortes, J. S. Denker, H. Drucker, I. Guyon, L. D. Jackel, Y. LeCun, U. A. Muller, E. Sackinger, P. Simard, and others, "Comparison of classifier methods: a case study in handwritten digit recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition, Vol. 3-Conference C: Signal Processing (Cat. No. 94CH3440-5)*, vol. 2. IEEE, 1994, pp. 77–82.

[12] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines."

[13] "NIPS Invited Talk Dropout: A simple and effective way to improve neural networks." [Online]. Available: https://neurips.cc/virtual/2012/invited-talk/3685

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[15] C. D. Manning, *An introduction to information retrieval*, 2009.

[16] C. Harris and M. Stephens, "A Combined Corner and Edge Detector," in *Procedings of the Alvey Vision Conference 1988*. Manchester: Alvey Vision Club, 1988, pp. 23.1–23.6. [Online]. Available: http://www.bmva.org/bmvc/1988/avc-88-023.html

[17] Y. Jia and T. Darrell, "Heavy-tailed Distances for Gradient Based Image Descriptors," in *Advances in Neural Information Processing Systems*, vol. 24. Curran Associates, Inc., 2011. [Online]. Available: https://proceedings.neurips.cc/paper/2011/hash/577bcc914f9e55d5e4e4f82f9f00e7d4-Abstract.html

[18] B. Fernando, E. Fromont, D. Muselet, and M. Sebban, "Supervised learning of Gaussian mixture models for visual vocabulary generation,"

Pattern Recognition, vol. 45, no. 2, pp. 897–907, Feb. 2012. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320311003098

[19] B. Liu, Y. Xia, and P. S. Yu, "Clustering through decision tree construction," in *Proceedings of the ninth international conference on Information and knowledge management*, 2000, pp. 20–29.

[20] F. Moosmann, B. Triggs, and F. Jurie, "Fast discriminative visual codebooks using randomized clustering forests," *Advances in neural information processing systems*, vol. 19, 2006.

[21] Y. Zhang, R. Jin, and Z.-H. Zhou, "Understanding bag-of-words model: a statistical framework," *International Journal of Machine Learning and Cybernetics*, vol. 1, no. 1, pp. 43–52, Dec. 2010. [Online]. Available: https://doi.org/10.1007/s13042-010-0001-0

[22] A. Vedaldi and A. Zisserman, "Efficient additive kernels via explicit feature maps," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 3, pp. 480–492, 2012.

[23] N. Cristianini and J. Shawe-Taylor, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[24] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Mar. 2015, arXiv:1502.03167 [cs]. [Online]. Available: http://arxiv.org/abs/1502.03167

[25] C. M. Bishop and N. M. Nasrabadi, *Pattern recognition and machine learning*. Springer, 2006, vol. 4, no. 4.

[26] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986, publisher: Nature Publishing Group UK London.

[27] L. Biewald, "Experiment Tracking with Weights and Biases," 2020. [Online]. Available: https://www.wandb.com/

[28] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning."

[29] "Writing VGG from Scratch in PyTorch | DigitalOcean." [Online]. Available: https://www.digitalocean.com/community/tutorials/vgg-from-scratch-pytorch

[30] G. Hinton, "Neural Networks for Machine Learning."

[31] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," Jan. 2017, arXiv:1412.6980 [cs]. [Online]. Available: http://arxiv.org/abs/1412.6980

[32] J. Duchi, E. Hazan, and Y. Singer, "Adaptive Subgradient Methods for Online Learning and Stochastic Optimization," *Journal of Machine Learning Research*, vol. 12, no. 61, pp. 2121–2159, 2011. [Online]. Available: http://jmlr.org/papers/v12/duchi11a.html

[33] "StepLR — PyTorch 2.7 documentation." [Online]. Available: https://docs.pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.StepLR.html

[34] "[1506.01186] Cyclical Learning Rates for Training Neural Networks." [Online]. Available: https://arxiv.org/abs/1506.01186

[35] "[1608.03983] SGDR: Stochastic Gradient Descent with Warm Restarts." [Online]. Available: https://arxiv.org/abs/1608.03983

[36] J. Bergstra, J. Bergstra, Y. Bengio, and Y. Bengio, "Random Search for Hyper-Parameter Optimization."

[37] "Local Invariant Feature Detectors: A Survey." [Online]. Available: https://ieeexplore.ieee.org/document/8187492

[38] S. RaoG and A. Sharma, "Cost Parameter Analysis and Comparison of Linear Kernel and Hellinger Kernel Mapping of SVM on Image Retrieval and Effects of Addition of Positive Images," *International Journal of Computer Applications*, vol. 73, no. 2, pp. 5–12, Jul. 2013. [Online]. Available: http://research.ijcaonline.org/volume73/number2/pxc3889517.pdf

[39] L. N. Smith, "Cyclical Learning Rates for Training Neural Networks," Apr. 2017, arXiv:1506.01186 [cs]. [Online]. Available: http://arxiv.org/abs/1506.01186

[40] ——, "A disciplined approach to neural network hyper-parameters: Part 1 – learning rate, batch size, momentum, and weight decay," Apr. 2018, arXiv:1803.09820 [cs]. [Online]. Available: http://arxiv.org/abs/1803.09820

[41] L. N. Smith and N. Topin, "Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates," May 2018, arXiv:1708.07120 [cs]. [Online]. Available: http://arxiv.org/abs/1708.07120

[42] M. Malinowski and M. Fritz, "Learnable pooling regions for image classification," *arXiv preprint arXiv:1301.3516*, 2013.

[43] J. Maas, E. Okafor, and M. Wiering, *The Dual Codebook: Combining Bags of Visual Words in Image Classification*, Nov. 2016.

APPENDIX A
CODE & PROJECT APPENDIX

A. Machine Specifications

Experiments were carried out across two machines:

1) **Local Machine:** AMD Ryzen 9 7945HX. Cores/Threads: 16 cores / 32 threads. Base Clock: 2.5 GHz. Boost Clock: Up to 5.4 GHz. NVIDIA RTX 4000 ADA SFF 20GB. 96GB SO-DIMM DDR5 5600mhz RAM. M.2 NVMe SSD 2TB. Operating System: Ubuntu 24.04 LTS. CUDA 12.8.
2) **Remote Machine:** DigitalOcean Cloud Droplet: 20vCPU, NVIDIA H100 80GB, 240GB RAM.

B. Project Source

The project source is available at https://github.com/jonathondilworth/uom-vision.