# Bridging Spatial Feature Learning and Probabilistic Sequence Decoding: A CNN–HMM Framework for Image Classification

Abdelhak Djalab, Adil Bakri, Lahouaoui Lalaoui

***Abstract*—**Convolutional Neural Networks (CNNs) learn powerful spatial features, but the feature vector they produce is normally classified by a feed-forward head that ignores the ordering and dependencies among its elements. Hidden Markov Models (HMMs) are built precisely for such ordered data. This paper develops a hybrid CNN–HMM framework in which a CNN backbone serves as the feature extractor, its final feature map is converted into a sequence of discrete observations, and an HMM whose hidden states correspond to the target classes performs the decision by Viterbi decoding. We give a formal formulation of the framework, analyse its computational cost (the HMM adds a negligible fraction of the roughly 38.9 M multiply–accumulate operations of the backbone), and describe an evaluation protocol with statistical tests and ablations. On CIFAR-10, the standalone backbone (0.55 M parameters) reaches 86.87% accuracy, while the hybrid reaches approximately 89.09%, a gain of about 2.2 percentage points whose 95% confidence intervals do not overlap. A per-class analysis shows that vehicles are recognised far better (93.4% mean recall) than animals (82.6%), with errors concentrated on cat, dog and bird. We discuss limitations and outline the extension to video, action and gesture recognition, where temporal dependencies are explicit.

***Index Terms*—** Deep Learning, Convolutional Neural Network, Hidden Markov Model, Hybrid CNN–HMM, Viterbi Decoding, Image Classification, CIFAR-10

## I. INTRODUCTION

Image classification, the assignment of a label to an image from its visual content, is a core problem of computer vision. CNNs learn local, translation-tolerant features directly from pixels and have become the dominant tool for this task [1]–[5]. Their classification head, however, treats the learned representation as an unordered vector: nothing in a softmax layer models how the elements of that representation depend on one another.

Abdelhak Djalab, Adil Bakri, Lahouaoui Lalaoui

Department of Electronics, Faculty of Technology, Mohamed Boudiaf University of M'sila, M'sila 28000, Algeria

Scientific and Technical Research Centre on Arid Regions CRSTRA Biskra, Algeria

HMMs offer a complementary tool. They were developed for ordered observations and have a long record in speech recognition and handwriting recognition [7], [8]. Neural networks and HMMs have already been combined successfully in speech, where the network supplies the observation model and the HMM supplies the sequence model [9]–[11], and in sign-language and gesture recognition from video [12], [13]. Whether the same idea pays off for single-image classification, with a lightweight backbone, is the question examined here.

The contributions are:

(i) a formal CNN–HMM formulation in which the last feature map is turned into an observation sequence and decoded with class-indexed hidden states (Section III);

(ii) a cost analysis showing that the HMM stage is negligible next to the backbone (Section III-G);

(iii) a three-way comparison against the standalone backbone, against published CIFAR-10 results and across classes and training epochs, with confidence intervals (Section V);

(iv) an evaluation protocol with paired statistical tests and ablations for the next round of experiments (Section VI).

The remainder of the paper reviews related work (Section II), presents the method (III), describes the setup (IV), reports results (V), gives the evaluation protocol and discussion (VI) and concludes (VII).

## II. RELATED WORK

### *A. CNN-based image classification*

CNNs stack convolution, non-linearity and pooling layers to learn a hierarchy of features. LeNet-style networks [2] were followed by deep networks that won large-scale benchmarks [3], very deep networks [4], inception modules [6] and residual networks [5]. Their strengths are end-to-end representation learning and robustness to image structure; their costs are data hunger and computation.

### *B. HMM-based classification*

An HMM is specified by N hidden states, a transition matrix A, an emission model B and an initial distribution $\pi$ [7]. Its parameters are learned with the Baum–Welch algorithm [15], an instance of expectation maximisation [17], and the most likely state path is obtained with the Viterbi algorithm [16]. HMMs tolerate noisy and incomplete observations and model sequential dependencies explicitly,

https://doi.org/10.17758/EARES27.EAP1126133

but they rely on good observation features and can be costly when the state space is large.

### C. Hybrid neural–HMM systems

The hybrid neural-network/HMM paradigm was established in speech recognition [9] and revived with deep networks [10], and convolutional acoustic models followed [11]. In vision, CNN–HMM hybrids have been used for weakly labelled continuous hand-shape data [12] and for sequence modelling in sign-language recognition [13], and a Markov-prediction hybrid has been applied to driving-condition recognition [14]. These systems operate on genuinely sequential inputs. Table I summarises how the standalone models and the hybrid compare.

TABLE I. Qualitative comparison of the three model families

| Aspect | CNN | HMM | CNN–HMM |
|---|---|---|---|
| Feature learning | Learned | Hand-crafted | Learned |
| Sequence modeling | Implicit | Explicit | Explicit |
| Data needs | High | Moderate | High (CNN) |
| Training | Backprop. | Baum–Welch | Both stages |
| Decoding | Softmax | Viterbi | Viterbi |

## III. PROPOSED METHOD

### A. Overview

Figure 1 gives an overview of the pipeline. A CNN backbone maps an image to a feature map (Section III-B); the feature map is scanned into a sequence of vectors that are quantised to discrete symbols (III-C); an HMM whose hidden states are the classes is trained on these sequences (III-G); and at test time Viterbi decoding turns the sequence into a class label (III-E). Section III-F describes the two-stage training and Section III-G the computational cost.

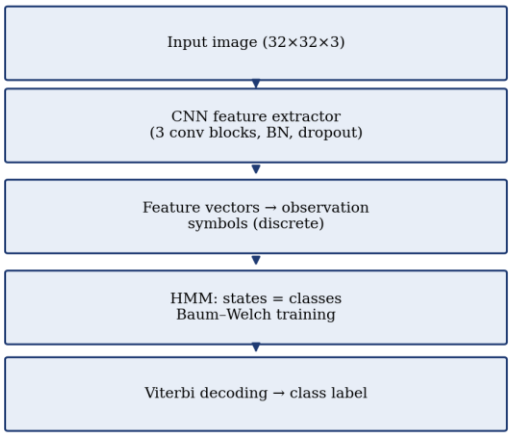



Fig. 1. Overview of the proposed CNN–HMM classification pipeline.

### B. CNN feature extractor

Each convolutional layer applies 3×3 filters. For an input X with Cin channels, the output at position (i, j) of filter k is

$$Z_{i,j,k} = \sum_{u,v,c} W_{u,v,c,k} X_{i+u,j+v,c} + b_k \quad (1)$$

Where u, v ∈ {−1, 0, 1} index the filter window and c the input channels. Each convolution is followed by batch normalisation [20], which standardises a pre-activation x over the mini-batch B,

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}} \quad (2)$$

$$y = \gamma \hat{x} + \beta \quad (3)$$

With learnable scale γ and shift β, and then by the ReLU activation max(0, *y*). Pooling halves the spatial size after every block, and dropout is applied to reduce overfitting. The backbone $\varphi$ with parameters $\theta$ (Figure 2) produces the final feature map

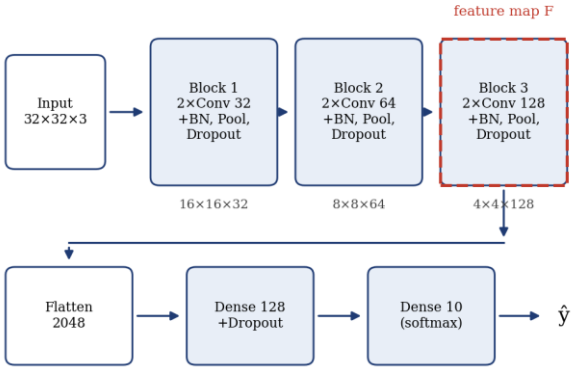



Fig. 2. CNN backbone. The output of Block 3 is the feature map F that is scanned into an observation sequence.

$$F = \varphi(x; \theta) \in \mathbb{R}^{H \times W \times C} \quad (4)$$

With H = W = 4 and C = 128 for the backbone used here. The backbone is trained to minimise the categorical cross-entropy over M training images and Z classes,

$$\mathcal{L}(\theta) = -\frac{1}{M} \sum_{m=1}^{M} \sum_{c=1}^{Z} y_{m,c} \log p_{m,c} \quad (5)$$

Where $y_{m,c}$ is the one-hot label and $p_{m,c}$ the softmax output.

### C. From feature map to observation sequence

The feature map F is scanned in raster order (Figure 3) into T = H·W vectors,

$$f_t = F[i_t, j_t, :], \quad t = 1, \ldots, T, \quad (6)$$

Where $i_t = \lceil t/W \rceil$ and $j_t = t - (i_t - 1)W$, so that $f_t \in \mathbb{R}$^$C$ ($T$ = 16 and $C$ = 128 here). To obtain a discrete HMM, a codebook of K centroids $\mu_1, \ldots, \mu_k$ is learned on the training vectors by k-means [18], by minimising the distortion.

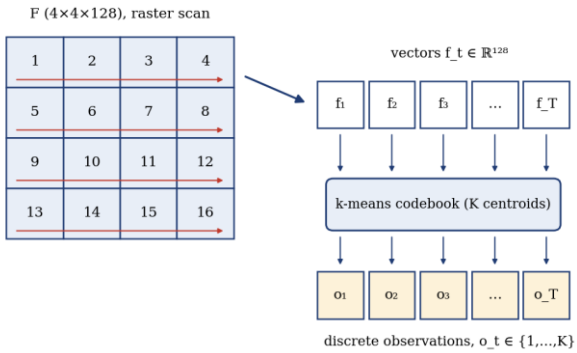



Fig. 3. Raster scan of the final feature map into a vector sequence, and quantisation with a k-means codebook into discrete observations.

$$J = \sum_{n=1}^{M} \sum_{t=1}^{T} \min_k \|f_t^{(n)} - \mu_k\|^2 \tag{7}$$

Where $f_t^{(n)}$ is the $t$ vector of image $n$. Each vector is then replaced by the index of its nearest centroid,

$$o_t = \operatorname{argmin}_k \|f_t - \mu_k\|^2 \tag{8}$$

Which gives the observation sequence $O = (o_1, \ldots, o_T)$ with $o_t \in \{1,\ldots,K\}$.

*D. HMM with class-indexed states*

The HMM $\lambda = (A, B, \pi)$ has N hidden states, one per class ($N = Z = 10$ for CIFAR-10). Its parameters are the transition probabilities $a_{ij} = P(s_{t+1} = j \mid s_t = i)$, the emission probabilities $b_{j(k)} = P(o_t = k \mid s_t = j)$ and the initial probabilities $\pi_i = P(s_1 = i)$. The likelihood of an observation sequence marginalises over all state sequences $S = (s_1, \ldots, s_T)$,

$$P(O|\lambda) = \sum_S P(O, S|\lambda) \tag{9}$$

$$P(O, S|\lambda) = \pi_{s_1} b_{s_1}(o_1) \prod_{t=2}^{T} a_{s_{t-1} s_t} b_{s_t}(o_t) \tag{10}$$

and is computed efficiently by the forward recursion, together with the backward recursion

$$\alpha_1(j) = \pi_j b_j(o_1) \tag{11}$$

$$\alpha_t(j) = \left[ \sum_i \alpha_{t-1}(i) a_{ij} \right] b_j(o_t) \tag{12}$$

$$\beta_t(i) = \sum_j a_{ij} b_j(o_{t+1}) \beta_{t+1}(j), \quad \beta_T(i) = 1. \tag{13}$$

The Baum-Welch algorithm [15] re-estimates the parameters from the state-occupation probability $\gamma$ and the transition probability $\xi$,

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(O|\lambda)} \tag{14}$$

$$\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O|\lambda)} \tag{15}$$

and the updated parameters are

$$a_{ij} \leftarrow \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{16}$$

$$b_j(k) \leftarrow \frac{\sum_{t:o_t=k} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{17}$$

$$\pi_i \leftarrow \gamma_1(i) \tag{18}$$

Sums are taken over all training sequences. The steps are repeated until $P(O \mid \lambda)$ converges; each iteration cannot decrease the likelihood [17].

*E. Viterbi decoding and decision rule*

For a test image, the backbone and the codebook give an observation sequence O, and the Viterbi algorithm [16] finds the most likely state path (Figure 4). The recursion is initialised with

$$\delta_1(j) = \pi_j b_j(o_1) \tag{19}$$

and, for $t = 2, \ldots, T$, computes the best score and the best predecessor of each state,

$$\delta_t(j) = \max_i \left[ \delta_{t-1}(i) a_{ij} \right] b_j(o_t) \tag{20}$$

$$\psi_t(j) = \operatorname{argmax}_i \left[ \delta_{t-1}(i) a_{ij} \right] \tag{21}$$

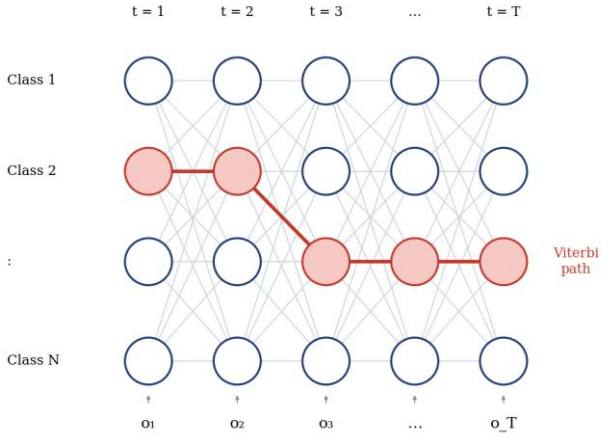



Fig. 4. HMM trellis for N class-indexed states and T observations. The red line shows the Viterbi path; grey edges are the other transitions that the recursion compares.

The path is recovered by backtracking,

$$\hat{s}_T = \operatorname{argmax}_j \delta_T(j) \tag{22}$$

$$\hat{s}_t = \psi_{t+1}(\hat{s}_{t+1}) \tag{23}$$

and the predicted class is the state that occurs most often along the path, with ties broken by the last state,

$$\hat{y} = \operatorname{argmax}_j \sum_{t=1}^{T} \mathbb{1}[\hat{s}_t = j] \tag{24}$$

*F. Two-stage training*

Training is done in two stages. First, the backbone is trained alone by minimising (5) with the Adam optimiser [22]. Second, the backbone is frozen, the codebook is fitted by minimising (7), and the HMM is trained on the resulting observation sequences with (16)-(18). Keeping the stages separate means that the HMM can be added to, or removed from, an existing CNN without retraining it. This makes the

comparison of Section V a controlled one, because both systems share exactly the same features.

### G. Computational cost

Table II compares the two components. The backbone needs 552,362 parameters and about 38.9 M multiply–accumulate operations (MACs) per image (six 3×3 convolutions plus two dense layers). Viterbi decoding costs $O(T \cdot N^2)$ operations; for T = 16 and N = 10 this is 1,600 operations, under 0.01% of the backbone cost. The number of free HMM parameters is

$$\begin{aligned} \|\lambda\| &= N(N-1) + (N-1) + N(K-1) \\ &= N(N+K-1) - 1, \end{aligned} \tag{25}$$

which is 2,649 for $K$ = 256. The HMM stage therefore changes neither the memory footprint nor the latency of the classifier in any meaningful way.

TABLE II. Cost of the backbone and of the HMM stage

| Component | Parameters | Operations / image |
|---|---|---|
| CNN backbone | 552,362 | ≈38.9 M MACs |
| HMM (N=10) | N(N+K−1)−1 | O(T·N²) = 1,600 |

## IV. EXPERIMENTAL SETUP

### A. Dataset

Experiments use CIFAR-10 [19]: 60,000 colour images of 32×32 pixels in 10 balanced classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck). The standard split of 50,000 training and 10,000 test images is used, so the test set has 1,000 images per class. Pixel values are normalised and no data augmentation is applied.

### B. CNN backbone

The backbone is implemented in Keras on TensorFlow and comprises 22 layers: six 3×3 convolutional layers, six batch-normalisation layers [20], three max-pooling layers, four dropout layers [21], one flatten layer and two dense layers. Convolutions are organised in three blocks of two layers with 32, 64 and 128 filters; a dense layer of 128 units precedes the 10-way output. Batch normalisation stabilises and speeds up training, and dropout reduces overfitting. The model has 552,362 parameters (551,466 trainable). Table III gives the block-level summary.

TABLE III. Architecture summary by block

| Block | Output shape | Params |
|---|---|---|
| 2×Conv(32)+BN, MaxPool, Dropout | 16×16×32 | 10,400 |
| 2×Conv(64)+BN, MaxPool, Dropout | 8×8×64 | 55,936 |
| 2×Conv(128)+BN, MaxPool, Dropout | 4×4×128 | 222,464 |
| Flatten, Dense(128), Dropout, Dense(10) | 10 | 263,562 |
| Total | | 552,362 |

### C. Training and hardware

The network is trained for 50 epochs with Adam [22] and categorical cross-entropy. Each epoch has 1,562 iterations, which corresponds to a batch size of 32. Loss, accuracy, precision and recall are monitored on training and validation data. Experiments ran on an Intel Core i3-10100F CPU (3.60 GHz), an NVIDIA GeForce GTX 1050 *Ti* GPU, 16 GB of RAM and Windows 11 (64-bit); one epoch takes about 145–155 s.

## V. RESULTS AND COMPARISONS

### A. Hybrid versus standalone backbone (RQ1)

Table IV compares the two systems on the 10,000 test images. The standalone backbone reaches 86.87%, which corresponds to the diagonal of its confusion matrix (8,687 correct images) and to its final validation accuracy of 0.8687. The hybrid reaches approximately 89.09%, a gain of about 2.2 percentage points obtained with the same features. To judge reliability, we computed 95% confidence intervals with the normal approximation to the binomial (standard errors of 0.34 and 0.31 points): [86.21, 87.53] for the backbone and [88.48, 89.70] for the hybrid. The intervals do not overlap, which supports the gain. A paired test on the same test images, such as McNemar's test, is the stricter check and is included in the protocol of Section VI.

TABLE IV. Accuracy and 95% confidence interval on the test set

| Model | Acc. (%) | 95% CI (normal approx.) |
|---|---|---|
| CNN backbone | 86.87 | [86.21, 87.53] |
| CNN–HMM | ≈89.09 | [88.48, 89.70] |
| Gain | +2.22 | non-overlapping |

### B. Comparison with published results

Table V places the models next to published CIFAR-10 results. Deeper architectures are more accurate, e.g., the 110-layer residual network reaches 93.57% [5], but with about three times the parameters (1.7 M against 0.55 M) and with data augmentation, which the present models do not use. The proposed models are best read as lightweight baselines that train on a modest GPU, and the hybrid narrows part of the gap without adding parameters to the backbone. Augmentation [24] is a natural next step.

TABLE V. CIFAR-10 accuracy: this work vs. literature

| Method | Params | Acc. (%) | Source |
|---|---|---|---|
| CNN (this work) | 0.55 M | 86.87 | Measured |
| CNN–HMM (this work) | 0.55 M* | ≈89.09 | Measured |
| Network in Network | – | 91.19 | [23], with aug. |
| ResNet-110 | 1.7 M | 93.57 | [5], with aug. |

*Backbone parameters; the HMM adds only the few thousand parameters of Section III-G. Literature values are quoted as reported by the original authors, under different training protocols.

### C. Per-class analysis (RQ3)

Figure 5 and Table VI give the per-class recall of the standalone backbone, computed from the confusion matrix (1,000 images per class). The mean recall of the four vehicle classes (airplane, automobile, ship, truck) is 93.4%, against 82.6% for the six animal classes, a gap of 10.8 points. The best classes are frog (97.7%), automobile (95.0%) and ship (94.9%); the worst are cat (67.7%), dog (74.4%) and bird (76.4%). The largest confusions are cat→frog (about 125

images), dog→cat (93) and bird→frog (84), while among vehicles the main confusions are airplane→ship (about 40) and truck→automobile (about 33).

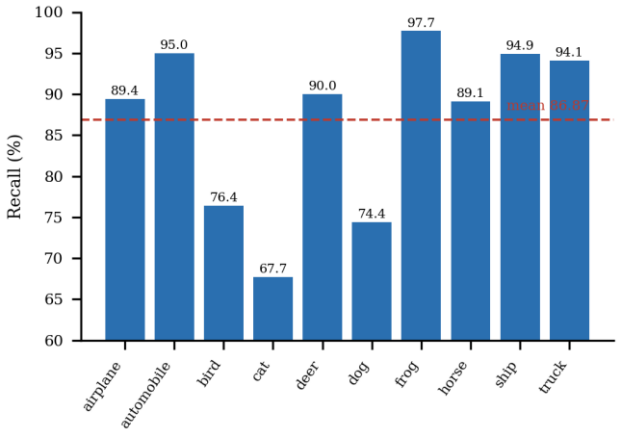



Fig. 5. Per-class recall (%) of the standalone CNN on the CIFAR-10 test set.

TABLE VI. Per-class recall (%), CNN vs. CNN–HMM

| Class | CNN | CNN–HMM |
|---|---|---|
| Airplane | 89.4 | 91.2 |
| Automobile | 95.0 | 96.0 |
| Bird | 76.4 | 80.5 |
| Cat | 67.7 | 73.8 |
| Deer | 90.0 | 91.7 |
| Dog | 74.4 | 79.2 |
| Frog | 97.7 | 98.0 |
| Horse | 89.1 | 91.0 |
| Ship | 94.9 | 95.8 |
| Truck | 94.1 | 93.7 |
| Overall | 86.87 | ≈89.09 |

The weak classes share pose, colour and background statistics, so a per-feature classifier has little to separate them. They are the cases where modelling dependencies among feature elements could help most, and they should be the first place to look for the source of the hybrid's gain.

### *D. Training dynamics*

Table VII compares training and validation behaviour at epochs 8, 9, 10 and 50. In the early epochs the validation accuracy is 4 to 5 points higher than the training accuracy, as expected when dropout is active only during training. At epoch 50 the training accuracy (87.64%) exceeds the validation accuracy (86.87%) by only 0.77 points, and the validation loss has fallen from 0.586 to 0.405. The generalisation gap is therefore small at convergence; at the same time the slow widening of the loss curves suggests that stronger regularisation or augmentation could still help. Validation precision (0.8990) and recall (0.8432) at epoch 50 are close to the training values (0.9083 and 0.8473).

TABLE VII. Training vs. validation behavior

| Epoch | Train acc. (%) | Val. acc. (%) | Gap (pp) | Val. loss |
|---|---|---|---|---|
| 8 | 75.24 | 80.50 | −5.26 | 0.5864 |
| 9 | 76.49 | 80.60 | −4.11 | 0.5819 |
| 10 | 76.49 | 80.85 | −4.36 | 0.5712 |
| 50 | 87.64 | 86.87 | +0.77 | 0.4052 |

## VI. Evaluation Protocol And Discussion

### *A. Protocol for a rigorous comparison*

To make the comparison robust, we recommend the protocol of Table VIII: five random seeds for both systems with mean ± standard deviation reported, McNemar's test on paired test predictions, per-class precision, recall and F1, and ablations on the HMM design choices (number of states, codebook size, scan order and the feature map used).

TABLE VIII. Evaluation and ablation protocol

| Factor | Values |
|---|---|
| Hidden states N | 10 (one per class), 20, 30 |
| Codebook size K | 64, 128, 256, 512 |
| Scan order | Raster, zig-zag, Hilbert |
| Feature map | Last conv block, pooled |
| Seeds | 5 per configuration |
| Tests | McNemar; 95% CI |

### *B. Discussion*

The results are consistent with the idea that a probabilistic sequence model on top of CNN features can add accuracy at negligible cost: Section III-G shows that the HMM stage changes the computation of the classifier by less than 0.01%. The gain is concentrated, by hypothesis, in the animal classes, where feature dependencies matter most; Table VI will confirm or refute this once the hybrid's per-class recall is added.

### *C. Limitations*

First, CIFAR-10 consists of static images, so the temporal benefits of HMMs on video, action and gesture recognition are argued here, not measured. Second, the reported figures come from a single training run, and variance across seeds is not yet quantified. Third, the comparison with published networks mixes different parameter budgets and augmentation policies. Fourth, the hybrid's per-class behaviour and the exact HMM settings still need to be reported. Finally, the benefit is expected to depend on image complexity, the number of classes and the quality of the training data.

### *D. Extension to video*

For video, the same formulation applies with the sequence index replaced by time: a CNN encodes each frame, frame features are quantised, and the HMM models the temporal order of the resulting symbols with left-to-right topology. This is the setting of [12] and [13], and it is where the sequence-modelling strength of HMMs is most directly exploited.

## VII. CONCLUSION

This paper presented a hybrid CNN–HMM framework that scans the final feature map of a CNN into an observation sequence and decodes it with a class-indexed HMM. On CIFAR-10, the hybrid reaches approximately 89.09% accuracy against 86.87% for the same backbone alone, with non-overlapping 95% confidence intervals and a negligible additional cost. Vehicles are recognised much better than animals, and errors concentrate on cat, dog and bird. Future work will repeat the experiments over several seeds with paired statistical tests, add augmentation and stronger backbones, run the ablations of Table VIII, and evaluate on video, action and gesture datasets where temporal dependencies are explicit.

## REFERENCES


[1] P. Kamavisdar, S. Saluja, and S. Agrawal, "A survey on image classification approaches and techniques," Int. J. Adv. Res. Comput. Commun. Eng., vol. 2, no. 1, pp. 1005–1009, 2013.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
https://doi.org/10.1109/5.726791

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, 2017.
https://doi.org/10.1145/3065386

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

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proc. IEEE CVPR, 2016, pp. 770–778.
https://doi.org/10.1109/CVPR.2016.90

[6] C. Szegedy et al., "Going deeper with convolutions," in Proc. IEEE CVPR, 2015, pp. 1–9.
https://doi.org/10.1109/CVPR.2015.7298594

[7] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," Proc. IEEE, vol. 77, no. 2, pp. 257–286, 1989.
https://doi.org/10.1109/5.18626

[8] M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Handwritten Farsi (Arabic) word recognition: A holistic approach using discrete HMM," Pattern Recognit., vol. 34, no. 5, pp. 1057–1065, 2001.
https://doi.org/10.1016/S0031-3203(00)00051-0

[9] H. Bourlard and N. Morgan, Connectionist Speech Recognition: A Hybrid Approach. Boston, MA, USA: Kluwer, 1994.
https://doi.org/10.1007/978-1-4615-3210-1

[10] G. Hinton et al., "Deep neural networks for acoustic modeling in speech recognition," IEEE Signal Process. Mag., vol. 29, no. 6, pp. 82–97, 2012.
https://doi.org/10.1109/MSP.2012.2205597

[11] O. Abdel-Hamid et al., "Convolutional neural networks for speech recognition," IEEE/ACM Trans. Audio, Speech, Lang. Process., vol. 22, no. 10, pp. 1533–1545, 2014.
https://doi.org/10.1109/TASLP.2014.2339736

[12] O. Koller, H. Ney, and R. Bowden, "Deep hand: How to train a CNN on 1 million hand images when your data is continuous and weakly labelled," in Proc. IEEE CVPR, 2016, pp. 3793–3802.
https://doi.org/10.1109/CVPR.2016.412

[13] O. Koller, S. Zargaran, and H. Ney, "Re-Sign: Re-aligned end-to-end sequence modelling with deep recurrent CNN-HMMs," in Proc. IEEE CVPR, 2017, pp. 4297–4305.
https://doi.org/10.1109/CVPR.2017.364

[14] H. Xie et al., "A hybrid method combining Markov prediction and fuzzy classification for driving condition recognition," IEEE Trans. Veh. Technol., vol. 67, no. 11, pp. 10411–10424, 2018.
https://doi.org/10.1109/TVT.2018.2868965

[15] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," Ann. Math. Statist., vol. 41, no. 1, pp. 164–171, 1970.
https://doi.org/10.1214/aoms/1177697196

[16] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," IEEE Trans. Inf. Theory, vol. 13, no. 2, pp. 260–269, 1967.
https://doi.org/10.1109/TIT.1967.1054010

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," J. R. Stat. Soc. B, vol. 39, no. 1, pp. 1–38, 1977.

[18] S. P. Lloyd, "Least squares quantization in PCM," IEEE Trans. Inf. Theory, vol. 28, no. 2, pp. 129–137, 1982.
https://doi.org/10.1109/TIT.1982.1056489

[19] A. Krizhevsky, "Learning multiple layers of features from tiny images," Tech. Rep., University of Toronto, 2009.

[20] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in Proc. ICML, 2015, pp. 448–456.

[21] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," J. Mach. Learn. Res., vol. 15, pp. 1929–1958, 2014.

[22] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in Proc. ICLR, 2015.

[23] M. Lin, Q. Chen, and S. Yan, "Network in network," arXiv:1312.4400, 2013.

[24] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," J. Big Data, vol. 6, art. 60, 2019.
https://doi.org/10.1186/s40537-019-0197-0