Supplementary Material: Mining Block I/O Traces for Cache Preloading with Sparse Temporal Non-parametric Mixture of Multivariate Poisson

Appendix A: Prediction Method

The prediction problem in the operational phase involves finding the best Z'_{t+1} using θ to solve equation 3.1. We describe a Viterbi like dynamic programming algorithm to solve this problem for Sparse-MVP emissions. (A similar procedure can be followed for MVP emissions).

We note that alternate strategies based on MCMC might be possible based on Bayesian inference for the variable under question. However, in the operation phase, the execution time becomes important and is required to be much smaller than ν, the slice length. Hence we explore the following dynamic programming based procedure that is efficient and runs in a small fraction of slice length ν.

At the end of the learning phase, we estimate the values of $\theta = \{\Lambda, \pi, b, \hat{\lambda}\}$, by obtaining $\Lambda, \hat{\lambda}, \pi$ as the mean of their posterior, and b by thresholding the mean of its posterior and use these as parameters during prediction. A standard approach to obtain the most likely decoding of the hidden state sequence is the Viterbi algorithm, a commonly used dynamic programming technique that finds

$$\{Z'^{*}_s\}_{s=1}^{t} = \underset{\{Z'_s\}_{s=1}^{t}}{\operatorname{argmax}}\ p(X'_1, ...X'_{t'}, \{Z'_s\}_{s=1}^{t})|\theta)$$

Let $\omega(t,k)$ be the highest probability along a single path ending with $Z'_t = k$. Further, let $\omega(t,k) = \max_{\{Z'_s\}_{s=1}^{t}} p(\{X'_s\}_{s=1}^{t}, \{Z'_s\}_{s=1}^{t})|\theta)$. We have

$$\omega(t+1,k) = \underset{k'=1,...,K}{\operatorname{argmax}}\ \omega(t,k')\pi_{k',k} SMVP(X'_{t+1};\theta)$$

Hence, in the standard setting of viterbi algorithm, having observed X'_{t+1}, the highest probability estimate of the latent variables is found as $Z'^{*}_{t+1} = \underset{1\le k\le K}{\operatorname{argmax}}\ \omega(t+1,k)$. However, the evaluation of MVP and hence the evaluation of the SMVP pmf involves exponential complexity due to integrating out the Y variables. While there are dynamic programming based approaches explored for MVP evaluation [17], we resort to a simple approximation. Let $\mu_{k,i} = \sum_{j=1}^{M} \lambda_{k,i,j} b_{k,i} b_{k,j} + (1-b_{k,i})\hat{\lambda}_j$, $i \in [M]$, $k \in [K]$. We consider $X_{t,i}|Z_t = k \sim Poisson(\mu_{k,i})$ when $X_t \sim SMVP(\Lambda_k, \hat{\lambda})$ (since the sum of independent Poisson random variables is again a Poisson random variable). Hence we compute $p(X_t|Z_t = k, \mu_k) = \Pi_{i=1}^{M} Poisson(X_{t,i}; \mu_{k,i})$.

In our setting, *we require finding the most likely* Z'^{*}_{t+1} *without having observed* X'_{t+1} to address our prediction problem from section 3. Hence we define the following optimization problem that tries to maximize the objective function over the value of X'_{t+1} along with the latent variables $\{Z'_s\}_{s=1}^{t+1}$.

$$\omega'(t+1,k) = \max_{\{Z'_s\}_{s=1}^{t+1}, X'_{t+1}} p(\{X'_s\}_{s=1}^{t+1}, \{Z'_s\}_{s=1}^{t+1})|\theta)$$

However, since mode of Poisson is also its mean,

$$\omega'(t+1,k) = Poisson(\mu_k|\mu_k) \max_{k=1,...,K} \omega'(t,k)\pi_{k,l} \quad (9.8)$$

From equation 9.8, we have a dynamic programming algorithm similar to Viterbi algorithm (detailed in algorithm 2).

Algorithm 2: Prediction Algorithm

Initial Iteration: Before X_1 **is observed**
$\omega'(1,k) = \pi_k^0 Poisson(\mu_k;\mu_k) \forall k$
$Z_1^* = Argmax_k\ \ \omega'(1,k)$
Initial Iteration: After X_1 **is observed**
$\omega(1,k) = \pi_k^0 Poisson(X_1;\mu_k) \forall k$
for $t = 2, \ldots T$ **do**
  Before X_t **is observed**
  $\omega'(t,l) = \max_k(\omega(t-1,k)\pi_{kl}) Poisson(\mu_l,\mu_l) \forall k$
  $Z_t^* = Argmax_k\ \ \omega'(t,l)$
  After X_t **is observed**
  $\omega(t,l) = \max_k(\omega(t-1,k)\pi_{kl}) Poisson(X_t,\mu_l) \forall k$
  $\Psi(t,l) = Argmax_k\ \ (\omega(t-1,k)\pi_{kl})$
Finding the Path $Z_T = Argmax_k \omega(t,K)$
for *Data points* $t = T-1, T-2 \ldots 1$ **do**
  $Z_T = \Psi(t+1, Z_(t+1))$

Appendix B: Inference Elaborated

In this section of supplementary material we discuss the inference procedure for DP-MMVP, HMM-DP-MMVP and Sparse-HMM-DP-MMVP more elaborately adding some details that could not be accomodated in the original paper. Our Collapsed Gibbs Sampling inference procedure is described in the rest of this section.

We first outline the inference for DP-MMVP model in section 10.1 , followed by the HMM-DP-MMVP, its temporal extension in section 10.2. Then, in section 10.3, we describe the inference for the Sparse-HMM-DP-MMVP model extending the previous procedure.

10.1 Inference : DP-MMVP: The existance of $Y_{t,j,l}$ latent variables in the MVP definition differentiates the inference procedure of an MVP mixture from

standard inference for DP mixtures. (The large number of $Y_{t,j,l}$ variables also leads to computationally expensive inference for higher dimensions motivating sparse modeling).

We collapse Λ variables exploiting the Poisson-Gamma conjugacy for faster mixing. The latent variables $Z_t, t \in [T]$, and $Y_{t,j,l}, j \leq l \in [M], t \in [T]$ require to be sampled. Throughout this section, we use the following notation: $Y = \{Y_t : t \in [T]\}$, $Z = \{Z_t : t \in [T]$ and $X = \{X_t : t \in [T]$. A set with a subscript starting with a hyphen$(-)$ indicates the set of all elements except the index following the hyphen.

Update for Z_t: The update for cluster assignments Z_t are based on the conditional obtained on integrating out G, based on the CRP[1] process leading to the following product.

$$p(Z_t = k|Z_{-t}, X, \beta, Y; \alpha, \bar{a}, \bar{b}) \propto p(Z_t = k|Z_{-t}; \alpha) f_k(Y_t)$$

$$\propto \begin{cases} n_k^{-t} f_k(Y_t) & \text{k} \in [K] \\ \alpha f_k(Y_t) & \text{k=K+1} \end{cases} \tag{10.9}$$

Where $n_{k,}^{-t} = \sum_{\bar{t} \neq t} \delta(Z_{\bar{t}}, k)$. The second term $f_k(Y_t) = p(Y_t, Y_{-t}|Z_t = k, Z_{-t}; \bar{a}, \bar{b})$ can be simplified by integrating out the Λ variables based on their conjugacy.
Let $S_{k,j,l} = \sum_{\bar{t}} Y_{\bar{t},j,l} \delta(Z_{\bar{t}}, k)$, for $j \leq l \in [M]$ and

$$F_{k,j,l} = \frac{\Gamma(\bar{a} + S_{k,j,l})}{(\bar{b} + n_k)^{(\bar{a} + S_{k,j,l})} \prod\limits_{\bar{t}: Z_{\bar{t}} = k} Y_{\bar{t},j,l}!} \tag{10.10}$$

By collapsing Λ, $f_k(Y_t) \propto \prod\limits_{1<=j<=l<=M} F_{k,j,l}$ (10.11)

Update for $Y_{t,j,l}$: This is the most expensive step since we have to update $\binom{M}{2}$ variables for each observation t. The Λ variables are collapsed, owing to the Poisson-Gamma conjugacy due to the choice of a gamma prior for the MVP.

In each row j of Y_t, $X_{t,j} = \sum_{l=1}^{M} Y_{t,j,l}$. To preserve this constraint, suppose for row j, we sample $Y_{t,j,l}, j \neq l$, $Y_{t,j,j}$ becomes a derived quantity as $Y_{t,j,j} = X_{t,j} - \sum_{p=1,p\neq j}^{M} Y_{t,p,j}$.

The update for $Y_{t,j,l}, j \neq l$ can be obtained by integrating out Λ to get an expression similar to that in equation 10.10. We however note that, updating the value of $Y_{t,j,l}$ impacts the value of only two other random variables i.e $Y_{t,j,j}$ and $Y_{t,l,l}$. Hence we get the following update for $Y_{t,j,l}$

$$p(Y_{t,j,l}|Y_{-t,j,l}, Z, \bar{a}, \bar{b}) \propto F_{k,j,l} F_{k,j,j} F_{k,l,l} \tag{10.12}$$

The support of $Y_{t,j,l}$, a positive, integer valued random variable, can be restricted as follows for efficient computation. We have $Y_{t,j,j} = X_{t,j} - \sum_{p=1,p\neq j}^{M} Y_{p,j} \geq 0$ Similarly, $Y_{t,l,l} = X_{t,l} - \sum_{p=1,p\neq l}^{M} Y_{p,l} \geq 0$. Hence, we can reduce the support of $Y_{t,j,l}$ to the following:

$$0 \leq Y_{t,j,l} \leq min\left((X_{t,j} - \sum_{p=1,p\neq l}^{M} Y_{p,j}), (X_{t,l} - \sum_{p=1,p\neq j}^{M} Y_{p,l}) \right) \tag{10.13}$$

10.2 Inference : HMM-DP-MMVP: The latent variables from the HMM-DP-MMVP model that require to be sampled include $Z_t, t \in [T]$, $Y_{t,j,l}, j, l \in [M], t \in [T]$, and $\beta = \{\beta_1, \ldots, \beta_K, \beta_{K+1} = \sum_{r=K+1}^{\infty} \beta_r\}$. Additionally an auxiliary variable m_k (denoting the cardinality of the partitions generated by the base DP) is introduced as a latent variable to aid the sampling of β based on the direct sampling procedure from HDP[16]. The latent variables Λ and π are collapsed to facilitate faster mixing. The procedure for sampling of $Y_{t,j,l}, j, l \in [M], t \in [T]$ is the same as that for DP-MMVP (eq: 10.12). Updates for m and β are similar to [4], detailed in algorithm 1. We now discuss the remaining updates.

Update for Z_t: The update for cluster assignment for the HMM-DP-MMVP while similar to that that of DP-MMVP also considers the temporal dependency between the hidden states. Similar to the procedure outlined in [16][4] we have:

$$p(Z_t = k|Z_{-t,-(t+1)}, z_{t+1} = l, X, \beta, Y; \alpha, \bar{a}, \bar{b})$$

$$\propto p(Z_t = k, z_{-t,-(t+1)}, z_{t+1} = l|\beta; \alpha, \bar{a}, \bar{b}) f_k(Y_t) \tag{10.14}$$

Where $f_k(Y_t) = p(Y_t, Y_{-t}|Z_t = k, Z_{-t}; \bar{a}, \bar{b})$. The first term can be evaluated to the following by integrating out π as

$$p(Z_t = k|Z_{-t,-(t+1)}, Z_{t+1} = l, \beta; \alpha) =$$

$$\begin{cases} (n_{z_{t-1},k}^{-t} + \alpha\beta_k) \frac{\alpha\beta_l + (n_{k,l}^{-(t)} + \delta(Z_{t-1},k)\delta(k,l))}{\alpha + n_{k,\cdot}^{-(t)} + \delta(Z_{t-1},k)} & \text{k} \in [K] \\ (\alpha\beta_{K+1}) \frac{\alpha\beta_l)}{(\alpha)} & \text{k=K+1} \end{cases} \tag{10.15}$$

Where $n_{k,l}^{-t} = \sum_{\bar{t}\neq t, \bar{t}\neq t+1} \delta(Z_{\bar{t}}, k)\delta(Z_{\bar{t}+1}, l)$. The second term $f_k(Y_t)$ is obtained from the equation 10.11.

10.3 Inference : Sparse-HMM-DP-MMVP: Sparse-HMM-DP-MMVP Inference is computationally less expensive due to the selective modeling of covariance structure. However, inference for Sparse-HMM-DP-MVPM requires sampling of $b_{k,j}, j \in [M], k = 1, \ldots$ in addition to latent variables in section 10.2 introducing challenges in the non-parametric setting that we discuss in this section. Note: Variables, η, Λ and $\hat{\lambda}$ are collapsed for faster mixing.

Update for $b_{k,j}$: The update can be written as a product:

$$p(b_{k,j}|b_{-k,j}, Y, Z) \sim p(b_{k,j}|b_{-k,j}) p(Y|b_{k,j}, b_{-k,j}, Z; \bar{a}\bar{b}) \tag{10.16}$$

By integrating out η, we simplify the first term as follows where $c_j^{-k} = \sum_{\bar{k}\neq k,\bar{k}=1}^{K} b_{k,j}$ is the number of clusters (excluding k) with dimension j active.

$$p(b_{k,j}|b_{-k,j}) \propto \frac{c_j^{-k} + b_{k,j} + a' - 1}{K + a' + b' - 1}$$

The second term can be simplified as follows in terms of $F_{k,j,l}$ as defined in equation 10.10 by collapsing the Λ variables and $\hat{F}_j$ obtained from integrating out the $\hat{\lambda}$ variables.

(10.17)
$$p(Y|b_{k,j}, b_{-k,j}, Z; \bar{a}\bar{b}) \propto \prod_{j\leq l\in[M]} F_{k,j,l}^{b_{k,j}b_{k,l}} \prod_{j\in[M]} \hat{F}_j$$

(10.18) Where $\hat{F}_j = \dfrac{\Gamma(\hat{a} + \hat{S}_j)}{(\hat{b} + \hat{n}_j)^{(\hat{a}+\hat{S}_j)} \prod_{t,j:b_{Z_t,j}=0} Y_{t,j,j}!}$

. And $\hat{S}_j = \sum_t Y_{t,j,j}(1 - b_{Z_t,j})$ and $\hat{n}_j = \sum_t (1 - b_{Z_t,j})$

Update for Z_t : Let $\mathbf{b^o} = \{\mathbf{b_k} : k \in [K]\}$ be the variables selecting active dimensions for the existing clusters. The update for cluster assignments Z_t, $t \in [T]$ while similar to the direct assignment sampling algorithm of HDP[16], has to handle the case of evaluating the probability of creating a new cluster with an unknown b_{k+1} .

The conditional for Z_t can be written as a product of two terms as that in equation 10.2

$$p(Z_t = k|Z_{-t,-(t+1)}, z_{t+1} = l, \mathbf{b^{old}}, X, \beta, Y; \alpha, \bar{a}, \bar{b}) \propto p(Z_t = k, Z_{-t,-(t+1),t+1} = l|\beta; \alpha, \bar{a}, \bar{b})$$
$$p(Y_t|Z_t = k, Z_{-t}, Y_{-t}, \mathbf{b^{old}}; \bar{\mathbf{a}}, \bar{\mathbf{b}}) \quad (10.19)$$

The first term can be simplified in a way similar to [4]. To evaluate the second term, two cases need to be considered.

Existing topic ($k \in [K]$) : In this case, the second term $p(Y_t|\mathbf{b^o} Z_t = k, Z_{-t}, Y_{-t}; \bar{a}, \bar{b})$ can be simplified by integrating out the Λ variables as in equation (10.17).

New topic ($k = K + 1$) : In this case, we wish to compute $p(Y_t|\mathbf{b^o}, Z_t = K + 1, Z_{-t}, Y_{-t}; \bar{a}, \bar{b})$. Since this expression is not conditioned on b_{K+1}, evaluation of this term requires summing out b_{K+1} as follows.

$$p(Y_t|\mathbf{b^o}, Z_t = K + 1, Z_{-t}, Y_{-t}; \bar{a}, \bar{b}) = \sum_{b_{K+1}} p(b_{K+1}|\mathbf{b^o}, \eta) p(Y_t|\mathbf{b^o}, b_{K+1}, Z_t = K + 1, Z_{-t}, Y_{-t}; \bar{a}, \bar{b})$$

Evaluating this summation involves exponential complexity. Hence we resort to a simple numerical approximation as follows. Let us denote $p(Y_t|\mathbf{b^o}, b_{K+1}, Z_t = K + 1, Z_{-t}, Y_{-t}; \bar{a}, \bar{b})$ as $h(b_{K+1})$ The above expression can be viewed as an expectation $E_{b_{K+1}}[h(b_{K+1})|\mathbf{b^o}]$. and can be approximated numerically by drawing samples of b_{K+1} with probability $p(b_{K+1}|\mathbf{b^o})$. We use Metropolis Hastings algorithm to get a fixed number S of samples using the proposal distribution that flips each element of b independently with a small probability $\hat{p}$. The intuition here is that we expect the feature selection vector for new cluster, b_{K+1} to be reasonably close to $b_{Z_t^{old}}$, the selection vector corresponding to the previous cluster assignment for this data point. In our experiments we set S=20 and $\hat{p}$=0.2 to give reasonable results.

We note that in [18], the authors address a similar problem of feature selection, however in a multinomial DP-mixture setting, by collapsing the b selection variable. However, their technique is specific to sparse Multinomial DP-mixtures.

Update for $Y_{t,j,l}$: The update for $Y_{t,j,l}$ is similar to that in section 10.1 with the following difference. We sample only $\{Y_{t,j,l} : b_j = 1, b_l = 1\}$ and the rest of the elements of Y are set to 0 with the exception of diagonal elements for the inactive dimensions. We note that for the inactive dimensions $\{j : j \in [M], b_{Z_t,j} = 0\}$, the value of $X_{t,j} = Y_{t,j,j}$ and hence can be set directly from the observed data without sampling.

For the active dimensions, $\{Y_{t,j,l} : b_j = 1, b_l = 1, j \leq l \in [M]\}$ we sample using a procedure similar to that in section 10.1 by sampling $Y_{t,j,l}, j \neq l$ to preserve the constraint $X_{t,j} = \sum_{l=1}^{M} Y_{t,j,l}$, restricting the support of the random variable in a procedure similar to section 10.1.

(10.20)
$$p(Y_{t,j,l}|Y_{-t,j,l}, Z, \bar{a}, \bar{b}, \hat{a}, \hat{b}) \propto F_{k,j,l} F_{k,j,j} F_{k,l,l} \prod_{1<=j<=M} \hat{F}_j$$

Appendix C: Experiment Details

11.4 Dataset Details: We perform experiments on publicly available real world block I/O traces from enterprise servers at Microsoft Research Cambridge [3]. They represent diverse enterprise workloads. These are about 36 traces comprising about a week worth of data, thus allowing us to study long ranging temporal dependencies. We eliminated 26 traces that are write heavy (write percentage > 25%) as we are focused on read cache. See Table 3 for the datasets and their read percentages. We present our results on the remaining 10 traces. We also validated our results on one of our internal workloads, NT1 comprising data collected over 24 hours.

We divide the available trace into two parts $\mathcal{D}^{lr}$

Algorithm 3: Inference: Sparse-HMM-DP-MMVP Inference steps(The steps for HMM-DP-MMVP are similar and are shown as alternate updates in brackets)

repeat
  for $t = 1, \ldots, T$ **do**
    Sample Z_t from Eqn 10.3 (Alt: Eqn 10.2)
    for $j \leq l \in [M]$ **do**
      if $b_{Z_t,j} = b_{Z_t,k} = 1$ **then**
        Sample $Y_{t,j,l}$ from Eqn 10.20 (Alt: Eqn 10.12)
        Set $Y_{t,j,j} = X_{t,j} - \sum_{\bar{j}=1}^{M} Y_{t,j,\bar{j}}$
        Set $Y_{t,l,l} = X_{t,l} - \sum_{\bar{l}=1}^{M} Y_{t,l,\bar{l}}$
  for $j = 1, \ldots, M, k = 1, \ldots, K$ **do**
    Sample $b_{k,j}$ from Eqn 10.16 (Alt: Set $b_{k,j} = 1_M$)
  for $k = K, \ldots, M, k = 1, \ldots$ **do**
    $m_k = 0$
    for $i = 1, \ldots, n_k$ **do**
      $u \sim Ber(\frac{\alpha\beta_k}{i+\alpha\beta_k})$, if $(u == 1) m_k ++$
  $[\beta_1\beta_2 \ldots \beta_K\beta_{K+1}]|m, \gamma \sim Dir(m_1, \ldots, m_k, \gamma)$
until *convergence* ;

that is aggregated into T_{lr} count vectors and $\mathcal{D}^{op}$ that is aggregated into T_{op} count vectors. We use a split of 50% data for learning phase and 50% for operation phase for our experiments such that $T_{lr} = T_{op}$.

Table 3: Dataset Description

Acro -nym	Trace Name	Description	Rd %
MT1	CAMRESWEBA03-lvm2	Web/SQL Srv	99.3
MT2	CAMRESSDPA03-lvm1	Source control	97.9
MT3	CAMRESWMSA03-lvm1	Media Srv	92.9
MT4	CAM-02-SRV-lvm1	User files	89.4
MT5	CAM-USP-01-lvm1	Print Srv	75.3
MT6	CAM-01-SRV-lvm2	User files	81.1
MT7	CAM-02-SRV-lvm4	User files	98.5
MT8	CAMRESSHMA-01-lvm1	HW Monitoring	95.3
MT9	CAM-02-SRV-lvm3	project files	94.8
MT10	CAM-02-SRV-lvm2	User files	87.6
NT1	InHouse Trace	Industrial	95.0

11.5 Design of Simulator: The design of our baseline simulator and that with preloading is described below.

Baseline: LRU Cache Simulator: We build a cache simulator that services access requests from the trace maintaining a cache. When a request for a new block comes in, the simulator checks the cache first. If the block is already in the cache it records a hit, else it records a miss and adds this block to the cache. The cache has a limited size (fixed to 5% the total trace size). When the cache is full, and a new block is to be added to the cache, the LRU replacement policy is used to select an existing block to remove. We use the hitrates obtained by running the traces on this simulator as our baseline.

LRU Cache Simulator with Preloading: In this augmented simulator, at the end of every $\nu = 30s$, predictions are made using the framework described in Section 3 and loaded into the cache (evicting existing blocks based on the LRU policy as necessary). While running the trace, hits and misses are kept track of, similar to the previous setup. The cache size used is the same as that in the previous setting.

11.6 Hitrate Values: Figure 4 in our paper shows a barchart of hitrates for comparison. In table 4 of this section, the actual hitrate values are provided comparing preloading with Sparse-HMM-DP-MMVP and that with baseline LRU simulator without preloading. We note that we show a dramatic improvement in hitrate in 4 of the traces while we beat the baseline without preloading in most of the other traces.

Trace	Preloading Sparse-HMM-DP-MMVP	LRU Without Preloading
MT1	**52.45 %**	0.02 %
MT2	**23.97 %**	0.1 %
NT1	**34.03 %**	03.66 %
MT3	**34.40 %**	6.90 %
MT4	**52.96 %**	42.12 %
MT5	**40.15 %**	39.75 %
MT6	**3.80 %**	3.60 %
MT7	**5.44 %**	5.22 %
MT8	**98.15 %**	98.04 %
MT9	**65.54 %**	65.54 %
MT10	**0.0 %**	0.0 %

Table 4: Comparing the hitrate with preloading with HMM-DP-MMVP and Sparse-HMM-DP-MMVP with hitrate of LRU without preloading. We note that Sparse-HMM-DP-MMVP beat the baseline showing dramatic improvement for the first four traces, and do well in comparison with the baseline for all the remaining traces.

11.7 Effect of Training Data Size: We expect to capture long range dependencies in access patterns when we observe a sufficient portion of the trace for training where such dependencies manifest. We show this by running our algorithm for different splits of train and test data (corresponding to the learning phase and the operational phase) for NT1 trace.

We observe that when we see at least 50% of the trace, there is a marked improvement in hitrate for the NT1 trace. Hence we use 50% as our data for training for our experimentation.

In a real world setting, we expect the amount of data required for training to vary across workloads. To adapt our methodology in such a setting periodic retraining to update the model with more and more data for learning as it is available is required. Exploring an

online version of our models might also prove useful in such settings.

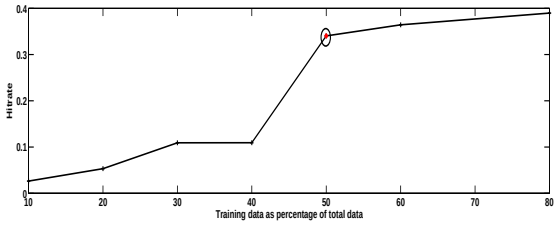

Figure 5: *Hitrate with increasing percentage of training data*