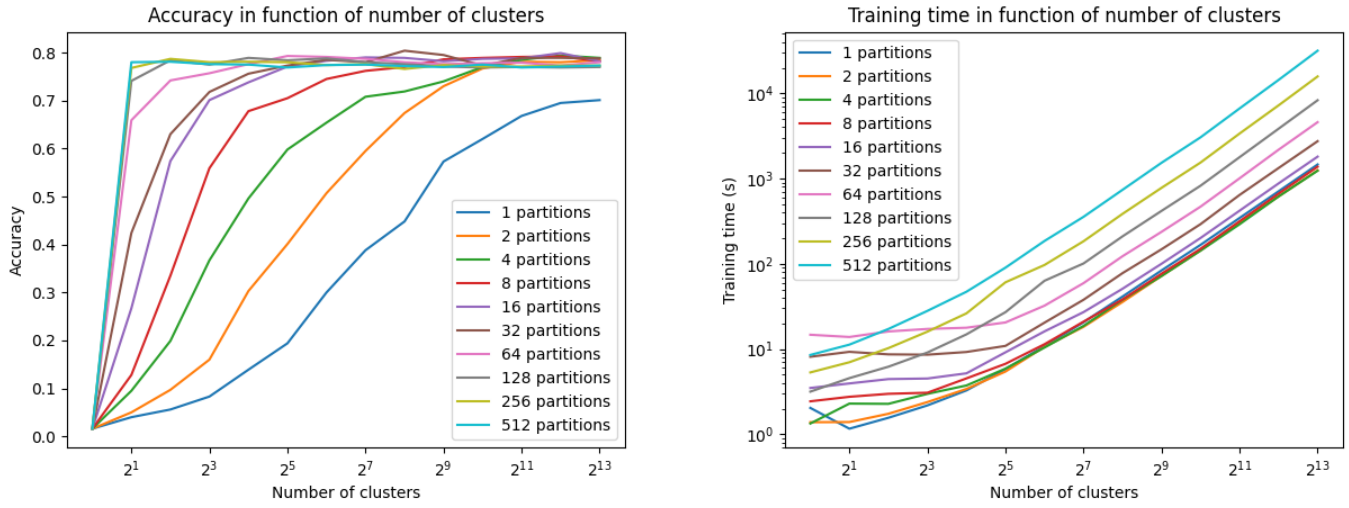
Assignment 3: BDAP [B-KUL-H00Y4A]

Q1 The approximate product quantization (PQ) method is faster than the naive nearest neighbors (NN) because it limits the amount of distance computations, which dominate execution time. Concretely: for naive NN N distances are computed per query in the case of N training examples. When using PQNN, the N training instances are mapped onto `nclusters` centroids $\ll N$. Only the distances between the query example and the `nclusters` must be calculated (per partition), hence the computation time drops significantly. Thus, although the search is still exhaustive, less memory has to be visited and only $\#partitions$ additions are required per distance calculation, rather than $\#features$ subtractions *and* multiplications*.

Q2 Figure 1a shows that accuracy increases with the number of partitions and clusters, as these refine the PQ's approximation. From figure 1b it can be seen that the execution time grows with the number of clusters and shifts upwards by a constant factor when increasing the number of partitions.



(a) Accuracy on `emnist_orig`

(b) Training time on `emnist_orig`

Figure 1: Effect of number of clusters and number of partitions on accuracy (1a) and execution time (1b) of `ProdQuanNN`

Q3 Purely based on accuracy, `npartitions` and `nclusters` should be chosen as large as is needed to achieve the maximum accuracy (cfr. plots 1a - 5a). This increases execution time, however. As such, a trade-off between execution time and accuracy should be made: choosing (`npartitions`, `nclusters`) $= (2^4, 2^5)$ for `emnist_orig` for example hits 72.5% accuracy ($\sim$ 96% of its peak accuracy), while still completing in under 10 seconds. Both hyperparameters should also be chosen sufficiently large to achieve a higher `retrieval_rate`.

Q4 The results of the experiments are shown in table 1. The `seed` and `k` hyperparameters were left on their default values of `1` and `10` respectively, while `n` was chosen $= 1000$ to have a representative sample size†.

		`time_and_accuracy` (`accuracy`, `time` [s])	`distance_error` (L^2 norm)	`retrieval`	Optimal hyperparameters (`npartitions`, `nclusters`)
`spambase`	`SKLearn`	(0.987, 0.1)	-	-	-
	`NumpyNN`	(0.987, 18)	-	-	-
	`PQNN`	(0.986, 4)	0.042	0.889	$(2^4, 2^4)$
`covtype`	`SKLearn`	(0.968, 10)	-	-	-
	`NumpyNN`	(0.968, 1625)	-	-	-
	`PQNN`	(0.952, 194)	0.073	0.868	$(2^4, 2^5)$
`emnist_orig`	`SKLearn`	(0.771, 4)	-	-	-
	`NumpyNN`	(0.771, 596)	-	-	-
	`PQNN`	(0.783, 177)	108.14	0.751	$(2^4, 2^5)$
`emnist`	`SKLearn`	(0.775, 5)	-	-	-
	`NumpyNN`	(0.775, 433)	-	-	-
	`PQNN`	(0.784, 130)	0.68	0.653	$(2^6, 2^6)$‡
`higgs`	`SKLearn`	(0.852, 4)	-	-	-
	`NumpyNN`	(0.852, 1091)	-	-	-
	`PQNN`	(0.859, 428)	0.027	0.849	$(2^5, 2^5)$

Table 1: Experiment results

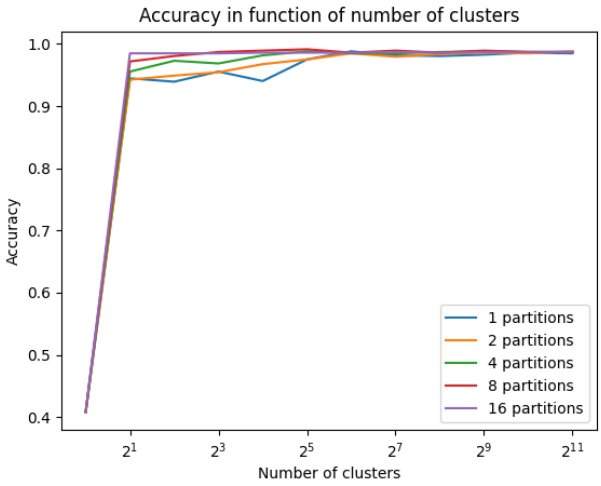
For completeness, the remaining plots on which the choice of the optimal hyperparameters was based are shown in 'Appendix: hyperparameter plots'. Implementation design choices are added as comments in the accompanying source code.

*`https://hal.inria.fr/inria-00410767/document/`

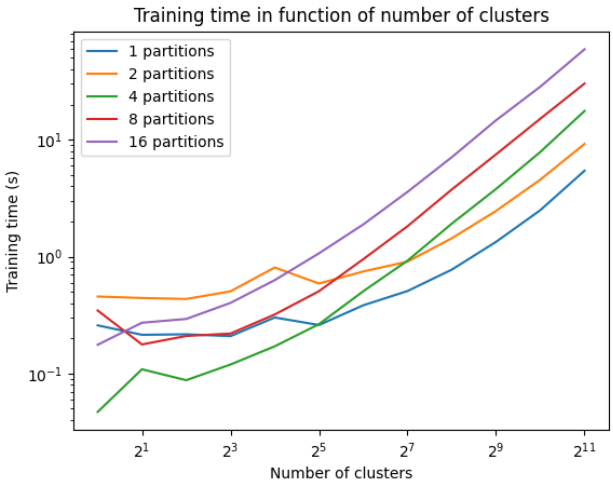
†`https://stats.stackexchange.com/a/172`

‡These hyperparameters can be chosen larger to increase `retrieval_rate`, but doing so will supersede `numpyNN`'s prediction time (due to the large number of features of `emnist`), nullifying product quantization's performance advantage.

Appendix: hyperparameter plots

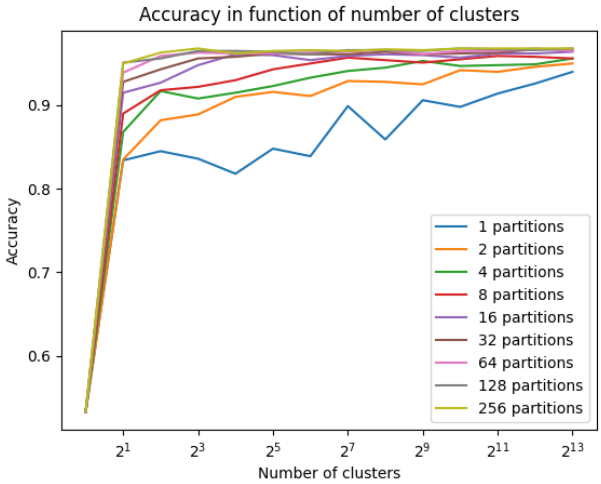


(a) `spambase` accuracy



(b) `spambase` execution time

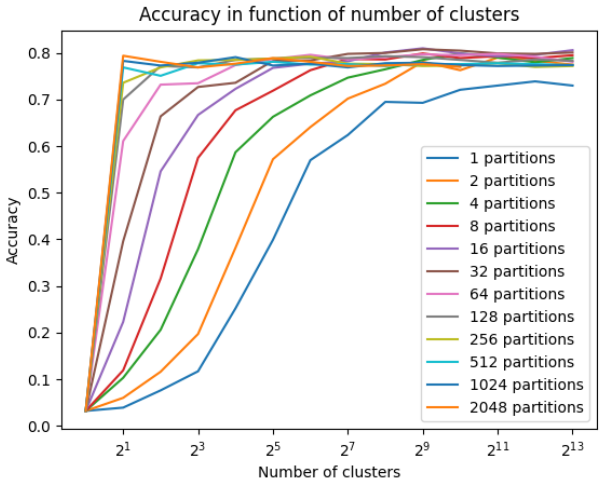
Figure 2: Effect of hyperparameters on `spambase` dataset



(a) `covtype` accuracy

(b) `covtype` execution time

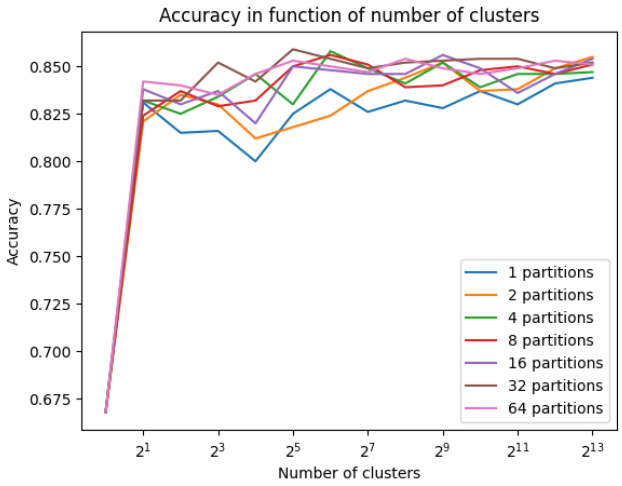
Figure 3: Effect of hyperparameters on `covtype` dataset



(a) `emnist` accuracy

(b) `emnist` execution time

Figure 4: Effect of hyperparameters on `emnist` dataset



(a) `higgs` accuracy

(b) `higgs` execution time

Figure 5: Effect of hyperparameters on `higgs` dataset