Mixture of Convolutional Neural Networks for Image Classification

By

RAVID COHEN

Supervisor

PROF. DAPHNA WEINSHALL

THE HEBREW
UNIVERSITY
OF JERUSALEM

Faculty of Computer Science and Engineering
THE HEBREW UNIVERSITY OF JERUSALEM

A dissertation submitted to the Hebrew University of Jerusalem as a partial fulfillment of the requirements of the degree of MASTER OF SCIENCE in the Faculty of Computer Science and Engineering.

SEPTEMBER 2018

Abstract

Two of the main issues that need to be considered when dealing with Mixture of Experts (ME) are how to partition the training data between the experts (partitioning method) and how to combine their output together (gating method). In a comprehensive series of experiments, we compared between many gating methods for mixture of convolutional neural networks to improve image classification. The methods we compared are known gating methods, and to a few of them we added some modifications. The results were mostly inconclusive, but they gave us insights on the complexity of ME. The following chapter introduces innovative approach for the partitioning phase. Specifically, it presents novel partitioning methods, where the labels are being partitioned instead of the images, in order to improve generalization of the experts. Finally, we explore the difficulties in creating a good partition for ME, such as balance-accuracy trade-off. The suggested architectures are inspiring and should be further investigated for other clustering purposes.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Mixture of experts (ME) is a popular combining method, in which a few models are combined to solve a task. ME, as well as other combining methods such as bagging, boosting, Stacked Generalization (SG) and AdaBoost improve performance in many fields of machine learning, especially in complex tasks [24, 25, 40]. The first configuration of ME was introduced more than 20 years ago [21] and is based on the divide-and-conquer principle. The problem space is divided between a few expert neural networks, supervised by a gating network (Figure 1.1). In one of the most basic setting, the final output of the ME is a weighted sum of the experts' outputs, where the weights vector is the output of the gating network. Even though ME has been developed a long time ago, it is still not a solved problem and many ME studies have been published in the last few years [28, 48]. Many strategies have been developed to divide the problem space between the experts (i.e. the partition method), while many others deal with the usage of the gating network (i.e. the gating method). In this work we investigated both ends of the problem with respect to large-scale image classification tasks.

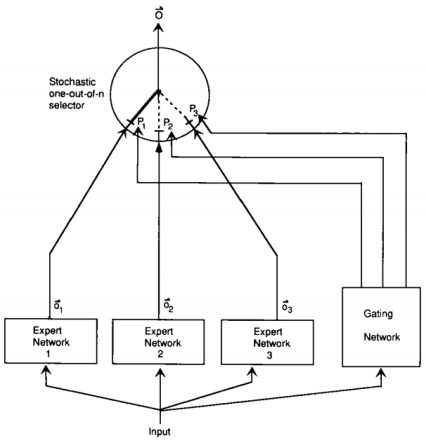


Figure 1.1: One of the first architectures suggested for ME [21] where the output is being chosen stochastically. Another common way to compute the final output of ME is $\sum_{k=1}^{K} P_k O_k$, where P_k is the k'th output of the gating network and O_k is the output of the k'th expert.

Recent image classification tasks of natural images have become increasingly challenging. For example, the iNaturalist 2017 competition [43], consists of 5089 organism species belonging to 13 super classes. While super classes are very different (e.g. fish and plants), species of the same super class may be visually very similar (Figure 1.2). For this reason, along with the low number of training images, this is one of the hardest image classification tasks. It is very intuitive to approach iNaturalist with ME since it is a large-scale classification task composed of many fine-grained classification tasks.

Figure 1.2: Images examples from iNaturalist competition 2017 show that species from the same genus may be visually very similar. Every row is a genus from a different super class and every image is a different species.

Data Partitioning Theoretical and empirical studies have shown that combining neural network experts is most effective when their errors are negatively correlated [16, 19, 41]. This can be achieved by randomly initializing the weights, using a variety of architectures, training with different optimizers or training the experts on different training sets [34]. Two popular methods for creating different training sets for experts are bagging [6] and boosting [32]. In bagging the training set is randomly sampled k times with replacement, and the experts are trained independently, while in boosting the distribution of the training set for one expert is adaptively changed based on the performances of the previous expert.

The conventional ME model [21] employs a special loss function to train the experts simultaneously while the problem space is partitioned stochastically. By doing so it encourages different

experts to learn different parts or aspects of the training data. Each expert becomes specialized on each subspace, and then a gating network learns the best partition according to the experts' efficiency in the different sub-spaces. Since the ME model was suggested, several studies have attempted to improve the stochastic partition of the problem space by modifying the loss function of the gating network [16, 20, 42]. Studies dating back to 2000 have proposed another approach, in which the input data is explicitly partitioned prior to the training [13, 15, 38]. In such methods, which generally achieve better performance, the input space is partitioned into more separable spaces and then each expert is trained on a pre-specified subspace.

As far as we know, all recent studies using a mixture of convolutional neural networks to improve image classification, divide the data explicitly. In particular, it is common to cluster the training images (with K-means for example) based on feature vectors obtained from an embedding layer of a pre-trained convolutional neural network (CNN) [9, 11, 14]. One disadvantage of this strategy is that it does not consider the class labels of the data, and may lead to unbalanced class partitioning into clusters [28]. To overcome this, some studies complete the training procedure with an end-to-end training [9], then all experts are eventually trained on all of the training images. This improves the results significantly, but is a very computationally expensive step, as the combined model may have millions of parameters. Our suggestion is to divide the **classes** between the experts instead of dividing the images. This way, we guarantee that each expert has enough training images for the classes it is supposed to be expert on.

Combining the Experts After training each expert on its own data subset, the final classification depends on some gating function. Given an image, such function give a score to each one of the experts that represent the probability that the input image belongs to the expert's subset of the data. There are various ways to define a gating function, and as far as we know no one have ever compared between all methods with respect to a mixture of CNNs. In chapter 3 we compare between three gating methods: the traditional gating model [21], the distance-based model [13, 14], and the more novel method of occupational probability [9]. The first two methods make use of the original base model, of which we use the embedding layer to cluster the images, while the latter depends on a confidence score obtained from the experts. Each of the above gating methods gives a score to each one of the experts, for a given test image. There are two main strategies to use the scoring for the final decision: fusion and selection. In the former strategy the final decision is made by considering the outputs of all experts where each one is weighted by its score, while in the latter strategy the final decision is made by aggregating the outputs of one or a few experts.

The rest of this thesis is organized as follows: Chapter 2 covers related work on ME and clustering algorithms, Chapter 3 describes our experiments for comparing the gating methods, Chapter 4 present our novel methods for data partitioning and Chapter 5 concludes and summarizes our work.

CHAPTER 2

RELATED WORK

Prior studies of ME are related to variable aspects of the model such as model selection, optimization algorithms and loss functions. In this work we investigate methods for explicitly partitioning the input data and methods of combining the output of the experts. Section 2.1 describes recent studies of mixture of convolutional neural networks for image classification. Section 2.2 introduce methods for explicit data partitioning and related clustering algorithms.

2.1 Mixture of CNNs for Image Classification

Ge et al. researched the usage of mixture of CNN experts and subset feature learning for fine-grained image classification [9–11]. In their work [11] they trained K experts on K non-overlapping subsets of Birdsnap database [4] to learn subset-specific features. The partition of the data they created by applying K-means on the embedding of the images. At test time they chose **one** expert (i.e. selection) by one of two gating methods: one is a gating CNN that was trained to predict the partition, and the second is by measuring the distance from the embedding of the test image to the clusters' centroids, and choosing the closest. They found the gating CNN to be more effective. Notice that they had to choose only one expert at test time because the experts were used for features extraction, not for classification so averaging their output does not really make sense. In their more recent study [9] they partitioned the data and trained the experts for classification in the same manner but they combined them differently. The combining was made by a **weighted sum** of the experts' output (i.e. fusion), using one of the following two gating methods: first was a gating CNN and the second was their main proposal, *occupational probability*, which supposed to represent the confidence of the experts without additional network (more on this in 3.1).

Ge et al. claim that their occupational probability method is more effective than gating network, but we suspect that their way of training the gate network does not give it a fair shot. Nonetheless, a few studies followed their suggestion and published successful results. [29] used a mixture of domain-specific CNNs for solving LifeCLEF 2016 plant task [12]. In this task, each sample is given by observations of few domains such as flower, leaf, branch etc. In their work they trained each one of the K experts on images of specific domain and combined them using occupational

probability. They were far from winning the competition but their final model did not required the organ (content) type to be specified in test time. Another study [30] combined a mixture of lightweight CNNs in a similar way for agricultural robotics purposes. They did not outperform a trained single Inception v3, but they reached a satisfying accuracy with one tenth of parameters.

While combining experts with occupational probability requires end to end training and therefore requires loading the whole combined model on a single GPU, [14] show that a hard mixture of experts can be efficient on large scale hashtag prediction task [39]. Their general strategy was parallel training of the experts and in test time choosing one expert, using a distance based model.

In chapter 3 we compare all three combining strategies (occupational probability, distance and gating network) alternating between selection and fusion.

2.2 Data Partitioning

Explicit partitioning of the input data into K pre-specified non-overlapping subsets has been shown to be effective when training a mixture of experts. Since it has been suggested, many researchers investigated various methods for such partitioning; naturally, most of them involve clustering. The partition method is crucial as it directly affect the performances of the experts in addition to the ability of the gating network to predict the right expert given a new input.

One innovative study [38] used Kohonen's Self Organizing Maps (SOM) [23] to cluster the input data prior to training the experts, and then the same SOM model is used as a gating network. SOM is a class of artificial neural networks (ANN) used for dimensionality reduction from a continuous input space to a discrete output space; SOM is therefore also useful for clustering. Kohonen's SOM is a feed forward network with a single layer in which the neurons are organized in a matrix; each neuron is fully connected to the input layer. A special update rule, based on the topology of the neurons and the similarity of the neurons' output to the input, is used to learn the mapping in an unsupervised manner. In 4.3 we suggest an architecture that reminds SOM. It is also used for clustering but in a semi-supervised manner and the weights update is based on entropy instead of Euclidean distance.

Another study presented a new algorithm for intensity modulated radiation therapy [13]. They clustered the input data directly using the fuzzy C-means algorithm to specify the data partition for the experts. The gating network was a radial-basis function (RBF) network, constructed using the cluster centroids as centers. Recent studies of mixture of CNNs for image classification also used the K-means algorithm to cluster the training images. In our experiments we noticed that often when following this method, the experts generalize worse than the base model. This way, images of one category may be assigned to more than one experts and as a result the experts have less images per category. Therefore, instead of clustering the images we turned to cluster the labels. Another way to look at it, is clustering the images under the constraint that two images of the same category must be in the same cluster. This leaded us to the field of constrained clustering.

Constrained Clustering is the employment of partially labeled data or pairwise constraints on some data points to aid clustering [5]. [45] introduced two types of constraints: either two instances must be in the same cluster, or they are known to be in different clusters. These two cases are called respectively *must-linked* and *cannot-linked* constraints. Most studies on this topic focused

on how to use a small number of pairwise constraints to guide the clustering algorithm towards a **more appropriate** data partitioning [3, 5, 7, 22, 26, 35, 45, 46]. This is done by modifying the clustering objective function [7, 17], enforcing constraints during the clustering process [45, 46], initializing based on labeled examples [3], or modifying the data according to the constrains and their implications [22]. Clustering images while forcing images with the same category label to be in the same cluster is technically constrained clustering but it is much more specific. We can formulate the problem by defining *must-linked* constraints on every successive pair of images of each class and feed it to one of the constrained-clustering algorithms, but it will be a great loss of information and will not be efficient. Some of the algorithms may converge very slowly due to the large number of constraints ([45] and [46]). Instead, we wish to leverage our additional knowledge: there are only *must-linked* constraints and every image has a constraint.

The only study that addressed a similar problem (as far as we know) is [35]. They notice that the *must-linked* constraints are transitive and therefore can form groups of points with *must-linked* constraints. They define a *chunklet* to be a set data points that are known to belong to the same cluster, and modify the Expectation Minimization (EM) algorithm accordingly. They suggest to take the sum in the 'E' step only over assignments which comply with the *chunklets* constraints. After analysis of their modification, they conclude that it is effectively equivalent to average each *chunklets* and treat it as a single points. Evaluation of the their modified EM shows faster convergence to a higher likelihood solution, compared to competitive methods.

In chapter 4 we suggest three methods for this specific task of *chunklets* clustering. The first is based on [35], and the other two are novel algorithms that utilizes special ANN architectures.

CHAPTER 3

COMBINING THE EXPERTS

We designed a thorough series of experiments to compare between all known gating methods for mixture of experts, in addition to some modifications of our own. This chapter describes the methods in detail and the adjustments we have made to them, followed by a description of the experiments and the results.

3.1 Methods

Progressive Transfer Learning In order to avoid overfitting, especially when combining deep neural networks with millions parameters, it is desirable to have as much data as possible. Similar to the suggestion of [11], we used ImageNet pre-trained model and fine-tuned it to the whole iNaturalist database as first step and then fine tuned it to our specific domain. The model that was trained on our target domain will be referred as *base model*.

***K*-means++** [1] The first step to constructing a mixture of K experts is to partition the training and validation data into K subsets. As [9] and [11] suggest, we applied k-means on features vectors that were obtained from the last fully connected layer of the *base model*. In our work, we used k-means++ [2] with multiple repetitions, and chose the clustering which minimizes the average silhouette score of the clustering. Given a clustering, the silhouette scorer of a sample x is:

$$s(x) = \frac{b(x) - a(x)}{\max(a(x), b(x))} \tag{3.1}$$

Where $a(x)$ is the average distance between x and all other points within the same cluster, and $b(x)$ is the average distance of x to all points in the closest cluster. Since computing the silhouette score requires $\Theta(n^2)$ distance computations, on big datasets we used *expected silhouette*. That is, randomly sample a subset of the data, calculate its silhouette score, repeat multiple times and return the mean silhouette score. We saw that the *expected silhouette* is very close to the actual silhouette score. Unlike [9] and [11], we have not reduced the dimension of the features vectors prior to the clustering.

[1] K-means++ is repetitions of K-means with smart initialization.

Training the Experts To achieve an expert on specific subset of images we fine tuned a pre-trained model on that subset. The question was whether to fine-tune the weights of the *base model* or to start with the ImagesNet weights. We examined the following four methods:

1. Fine tune ImageNet weights, exactly as we trained the base model.
2. Fine tune the base model with low learning rate.
3. Randomly initialize the last layer of the base model, and fine tune it with low learning rate.
4. Same as 3, with high learning rate.

We trained multiple expert models with each one of the above settings and measured their final validation accuracy. We found that the highest generalization is achieved by the last method. Evaluation of the experts can be done on each one individually, or on all of them together by calculating their mean performance. *Mean expert accuracy* is the weighted average of experts' accuracies, where weights are the ratio of the expert's subset size.

Gating Network In our work, gating network is a CNN that learns directly the initial partition of the data. Let $E(x)$ denote the output of the embedding layer on some input image x. Our gating model applies the following to $E(x)$:

1. l2-norm. (Since K-means is performed on normalized features vectors)
2. Dropout layer.
3. Dense layer to the number of experts.
4. *Softmax* activation.

The k'th output of the gating network represents the confidence score for the k'th expert. This is the only gating method that requires an additional training. How to train it? We examined the following approaches:

- End-to-end fitting vs. fitting only the last fully connected layer of the gating network.
- With or without $l2$-normalization.
- Linear SVM classifier of the feature vectors.

We trained gate models using the different approaches and compared their validation accuracy. We found that fitting only the last layer of the gate model generalizes much better then end-to-end fitting and Linear SVM classifier (see Table 3.1). Therefore, this process is relatively fast.

Table 3.1: How to train the gate model

	End-to-end	Last layer	Linear SVM
W/o $l2$-norm	92.5	95.3	N/A
With $l2$-norm	92.0	**97.4**	95.8

Table 3.1 Comparing different ways to train the gate model. Best generalization achieved when using $l2$-normalization of the embedding layer and training only the last fully connected layer of the model.

Occupational probability As [9] suggest, we examined the following occupational probability based on the output of the experts alone, without using an additional model:

$$\alpha(x)_k = \frac{\exp(M(x)_k)}{\sum_{k'=1}^{K} \exp(M_{k'}(x))} \tag{3.2}$$

Where M_k is the best classification results of the k'th expert. That is, let O_k denote the output of the k'th expert, prior to the *softmax* activation, then:

$$M(x)_k = \max_{n=1...N} O_{k,n}(x) \tag{3.3}$$

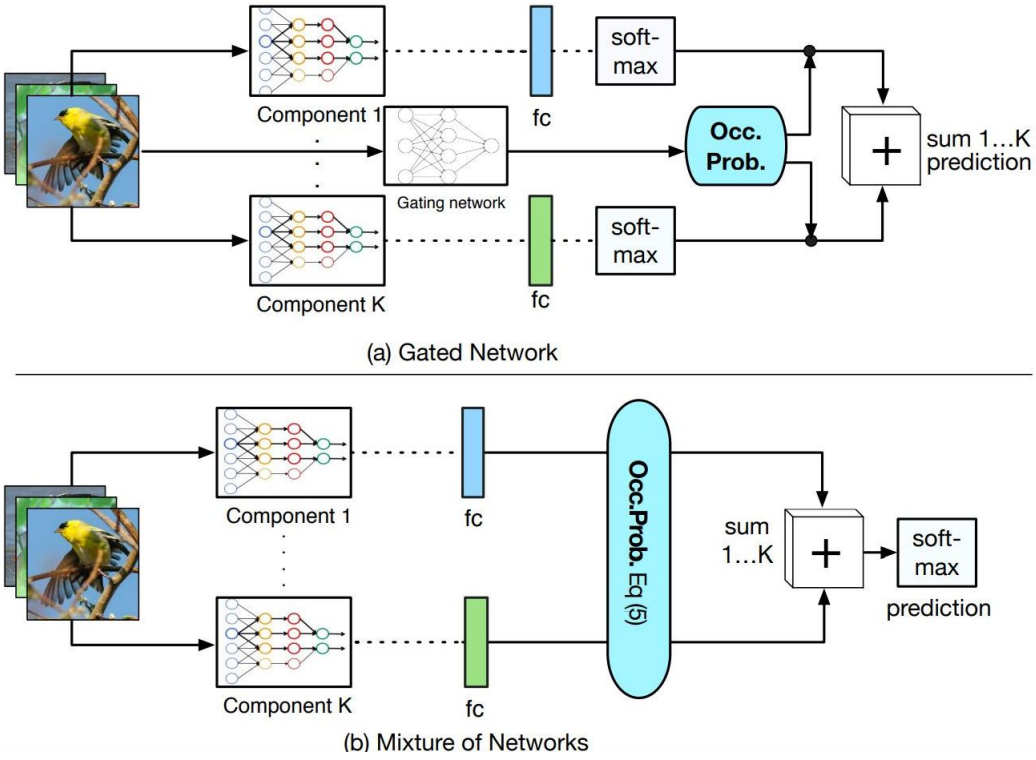


Figure 3.1: Architecture of mixture of experts, combined with gating network (a) and occupational probability (b). Taken from [9]

Distance Based Gating Inspired by [13] and [14], we constructed a gating network based on the Euclidean distance between the embedding and the clusters centroids. Let $c_1, ..., c_K$ denote the clusters' centroids. Then given an input image x, we can calculate the squared Euclidean distance between its embedding to the k'th centroid:

$$dist^2(x)_k = ||E(x) - c_k||_2^2 \tag{3.4}$$

And the confidence score of the k'th expert is the normalized Radial basis function (RBF):

$$\alpha(x)_k = \frac{\exp(-dist^2(x)_k)}{\sum_{k'=1}^{K} \exp(-dist^2(x)_{k'})} \tag{3.5}$$

Combining Gating Methods In this work we tried to use a combination of two gating methods. Let $\beta(x), \gamma(x)$ be two confidence distributions of the experts, of two different gating methods. We suggest to combine them by the following:

$$\alpha(x)_k = p\beta(x)_k + (1-p)\gamma(x)_k \tag{3.6}$$

Where $p \in [0,1]$ is a trainable parameter.

Fusion vs Selection Given a confidence distribution of the experts, we can choose the one with the highest score (selection) or sum the outputs of all experts with their confidence scores as weights (fusion). We can generalize the two strategies by defining a *selection amount* parameter $s \in \{1,...,K\}$. Then given a confidence distributions $\alpha(x)$, zero out the scores that are not in the top s and normalize the remaining scores to form a stochastic vector. That is:

$$\alpha'(x)_k = \begin{cases} \frac{\alpha(x)_k}{sum(sort(\alpha(x))[K-s,...,K])} & \text{if } k \in argsort(\alpha(x))[K-s,...,K] \\ 0 & e.w \end{cases}$$

ME Ensemble The traditional ensemble method is to sum or average the output of more than one model, where each one of them was trained on the whole database. In ME ensemble we sum the output of ME model with the base model.

3.2 Experiments and Results

Dataset We performed all of the experiments in this chapter on the 2017 iNaturalist competition database [43] that contains images of over 5000 species of animals and plants from all over the world. The database consists of 13 super categories which are visually different (mammals, birds, plants etc.), while within the subcategories the species can be very similar. Due to disambiguated data and cases where more than one species happens to be in the same image, the evaluation metric used in the competition is top-5 accuracy. The current state of the art is an error rate of 0.05, using an ensemble of different CNNs. Considering the size of this database we decided to use only the Aves super category which has the highest number of image per species (over 215,000 training images of 964 bird species).

Experiment We designed a thorough series of experiments that compare between all three gating methods and all their combinations with selection vs. fusion, using variable *selection amount*, including a combination of more than one gating methods. Firstly, we fine-tuned Inception V3 [37] from imageNet pretrained weights to iNaturalist aves images. This from now on will be referred as *base model*. We then applied kmeans++ with $K = 4$ on the embedding (*pool_3* layer) of the training images to achieve a partition. The validation images were assigned to the clusters base on distance from clusters centroids. We fine-tuned 4 copies of the base model on each one of the subsets to achieve the experts. Another base model was fine-tuned to predict the partition, as a gating network. Finally, we combined the experts to form a mixture of experts in different manners as described above. For each one of the methods we evaluated the following:

- Performances of the gating method in predicting the correct expert.

- Initial performances of the ME, before any farther fitting.
- Final performances of the ME, after an end-to-end fitting the whole model, for 16 epochs.

Implementation The base model, gate model and experts were trained using SGD optimizer with momentum of 0.9 and exponential learning rate decay of 0.94 every 4 epochs. For the final end to end training of the ME model we used Adam optimizer with $\epsilon = 0.1$, learning rate of 0.002 and learning rate decay on plateau.

Gating method	Gating accuracy	Selection amount	ME accuracy	
			initial	final
Gating Network	92.5	1	77.2	-
		2	78.0	78.6
		3	**78.1**	-
		4	**78.1**	78.7
Occprob	62.8	1	72.1	78.4
		2	71.3	79.0
		3	71.1	79.0
		4	71.0	79.0
Distance	100	1	77.3	77.5
		2	-	-
		3	-	-
		4	70.9	70.2
Gate + Occprob	89.8	1	-	-
		2	75.1	79.1
		3	-	-
		4	74.6	79.2
Occprob + Base (ME Ensemble)	-	4	-	**80.2**

Table 3.2 Comparison of different gating methods for ME. Gating accuracy is the top-1 accuracy of the gating model in predicting the correct expert. ME accuracy initial and final are the top-5 accuracies of the ME model prior to farther training and after 16 epochs of end to end training respectively. Each complete row of this table requires about 120 GPU hours.

Results Table 3.2 presents the main results of our experiments, which required approximately 1200 GPU hours. We were quite disappointed to see that none of the ME models outperforms the base model (Figure 3.2). Even though we replicated almost the whole process of [9], we did not obtain the same results, but one should keep in mind that [9] did not mention in their paper how many epochs of end-to-end training were required for the results they presented. Nonetheless, we can infer some conclusions from the order of the results. The gated ME provided the best initial results. This is very important since end to end training of the ME is computationally expensive (6-8 hours per epoch). ME with occupational probability on the other hand shows poor initial performances but outperforms the gated ME after an end to end training. The distance based ME shows the worst results but it is interesting to see that this model clearly prefers selection over

fusion, which justify the work of [14], while all of the other models do not seem to be affected by it significantly. Combining gate model and occupational probability together gives better accuracy than each one of them separately, not by far. In another test (see ME Ensemble in 3.1), we combined the base model with occupational probability ME mode and achieved top-5 accuracy of 80.2%, similar to ensemble of 5 base models (Figure 3.2).

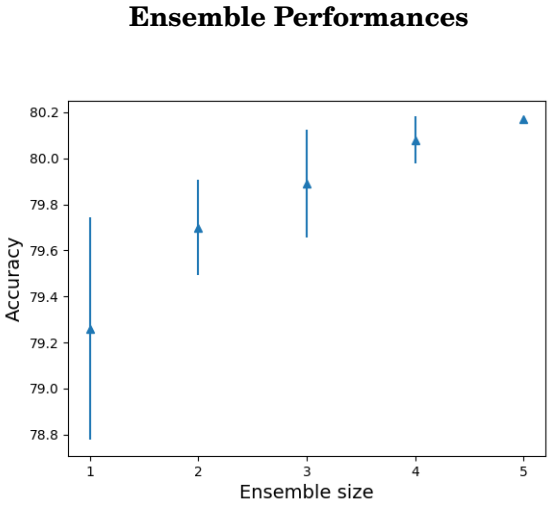


Figure 3.2: We trained 5 base models and tested all combinations of ensembles, averaging the outputs of 2, 3, 4 and 5 models.

To understand better the insufficiency of the ME we evaluated the base model on the different data partitions and compared it to the performances of the experts (Table 3.2). We noticed that all four experts perform poorly on their subset of data, compared to the base model. This observation is quite disturbing as it contradicts the idea of an expert. One possible reason for this observation is the fact that we did not reduced the dimension of the embedding prior to the clustering and therefore the features vectors contains many features which are irrelevant for classification. In the next section we expand more on this subject.

Table 3.2: Expert vs. Base Model

Cluster	Base Model	Expert
0	79.2	76.8
1	81.4	78.3
2	84.2	82.0
3	71.1	67.8
Total	79.1	77.3

Table 3.2 Top-5 accuracy of the experts on their clusters of images comparing to the base model.

3.3 Dimensionality Reduction

Clustering a high dimensional data has well known issues [1]. Besides that, we suspect that the 2048 dimensional embedding does not capture the very essential features required for classification,

but possibly more unnecessary features that remain from the transfer learning. In addition, as [11] showed, some features are related to pose and we do not want that to affect the partitioning of the data to experts. This section presents our method for dimensionality reduction and repetition of the previous experiment in lower dimension.

3.3.1 Methods

Stacked Auto Encoders To reduce the dimension of the embedding layer we added to the trained base model stacked blocks of auto encoders (SAE), each one reducing the dimension by half. An auto encoder (AE) block composed by the following layers:

1. Dropout Layer.
2. Dense layer, reducing the dimension by half.
3. ELU activation.
4. Batch normalization.

Inspired by [44] and [47] we trained the SAE model **layer by layer**. Unlike [44] and [47], our SAE was not trained for reconstruction, but for classification. At every step in the training process we added a new AE block, followed by a dense layer and a softmax activation (for classification). We froze the whole graph, except for the new added layers and trained it until convergence. We then removed the classification layer and continued to the next step (training the next AE block). To complete the training, we trained the whole SAE model end to end with lower learning rate until convergence. We compared few different architectures and training algorithms and present here the one that gives best generalization (Table 3.3).

Table 3.3: SAE Architecture and Training

	Top-5 Accuracy
No BN, end to end training	78.71±0.08
BN before activation, end to end training	78.74±0.04
BN after activation, end to end training	79.13±0.06
BN after activation, layer-wise training	**79.34**±0.05

Table 3.3 Performances of SAE model with different settings. We compared models with batch normalization before and after the non linear activation function or without BN at all. Moreover, we compared end to end training and layer-wise training.

3.3.2 Experiments and Results

Experiment We trained 9 SAE models with 1-9 AE layers. After training an SAE model we followed the same procedure as in the previous section with the basic settings: We applied K-means++ with $k = 4$ on the embedding (this time with lower dimension) to form a partition of the data, trained a gate model and experts accordingly and combined them together to form the ME. We compared ME with gate model and occupational probability with *selection amount* = *4*.

Results Table 3.4 presents the performances of each module in each one of the experiments, which took a total of more than 2400 GPU hours. Again here, none of the ME models outperforms the base model. In particular, the SAE with one encoding layer has similar performances to ensemble of 5 models (Figure 3.2). We do not notice a dramatic effect of the dimensionality reduction on the performances of the expert, gate and ME. However, as in the previous experiment we see that the gated ME has better initial accuracy in all cases but after end to end training the occprob ME outperforms it, not by far.

Table 3.4: SAE results

AE Layers	Embedding Dimension	SAE Model Accuracy	Mean experts Accuracy	Gate Accuracy	Gated ME		Occprob ME	
					initial	final	initial	final
0	2048	79.1	77.3	97.4	77.8	78.3	71.0	**79.0**
1	1024	**80.2**	76.1	96.0	77.6	77.9	64.7	78.1
2	512	80.1	76.2	94.9	77.5	-	-	-
3	128	79.8	76.4	95.3	77.7	-	-	-
4	64	79.2	75.9	93.4	77.3	77.3	67.8	78.0
5	32	77.8	77.0	96.2	77.9	-	-	-
6	16	73.7	77.5	96.4	77.8	77.7	73.6	78.6
7	8	55.3	76.7	96.1	77.0	-	-	-
8	4	27.1	76.4	95.6	76.4	-	-	-
9	2	8.2	77.4	97.7	76.7	76.2	71.4	78.6

Table 3.4 Comparison of 10 models with different number of auto encoding layers which decrease the dimension of the embedding. The table presents the accuracy of the SAE model (top-5), the mean accuracy of the experts (top-5), the accuracy of the gating network (top-1) and the accuracy (top-5) of the ME models before and after end to end training. Each complete row in this table required approximately 360 GPU hours.

Analyzing The Gated ME Since end to end training is very expensive we decide to invest our efforts in improving the gated ME. To analyze the gated ME we look at the probability of predicting the correct label of an image as a joint probability of the two event: A) the gate model predicts the correct expert and B) the expert predicts the correct label. Figure 3.3 shows that multiplying the performances of the gate model and the experts gives a good estimation of the performances of the gated ME. This fact is not surprising and it gives us a motivation to optimize each one of the modules separately.

Recall the generalization problem of the experts (Table 3.2). Another reason for this observation might lay in the fact that the partition of the data was over the images and not over the categories. Therefore one category can be distributed within two or more experts and as a result the experts are being trained on less images-per-category. To examine this thought we calculated the distribution of images for each category across the k subsets. That is, given that two images are from the same category, what is the probability that they are in the same cluster. Let $S = \{(x_i, y_i)\}_{i=1}^{n}$ denote the training images and corresponding category labels and let $c(x)$ denote the cluster (subset) to which x is assigned to. Then the *category-clusters-distribution (ccd)* of clustering c on dataset S defined

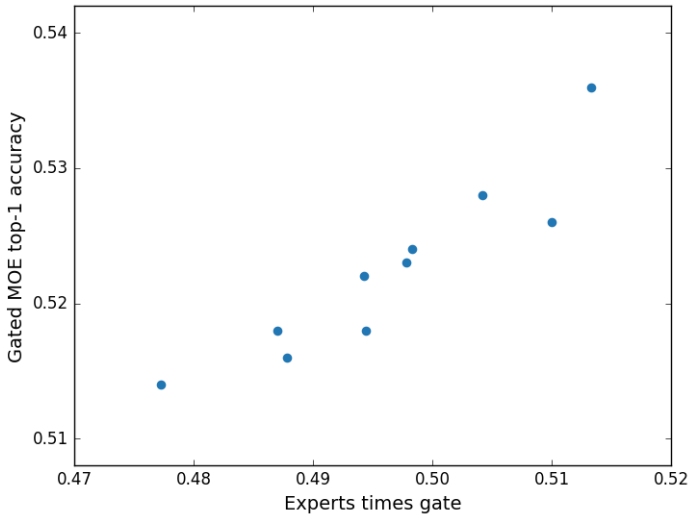


Figure 3.3: For each one of the SAE models we computed the mean expert top-1 accuracy times the top-1 accuracy of the gate model and plotted it along with the top-1 accuracy of the gated ME model.

by:

$$ccd(S,c) = P_S[c(x_i) = c(x_j) | y_i = y_j] = \frac{|\{(x_i, x_j \in S : y_j = y_i, c(x_i) = c(x_j)\}|}{|\{(x_i, x_j \in S : y_j = y_i\}|} \tag{3.7}$$

We compared the ccd of the clustering that were obtained by the different SAE models with the mean accuracy of their experts (Figure 3.4). Even though the points do not sit on a straight line, it convinced us that increasing the ccd may increase the generalization of the experts.

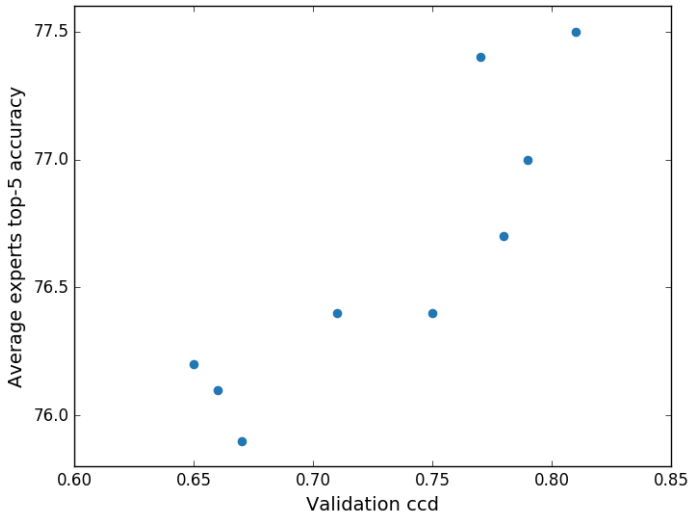


Figure 3.4: For each one of the partitions we computed the *category-clusters-distribution* of the validation set and plotted it along the mean accuracy of the experts on this partition.

Chapter 4

Constrained Partitioning

Motivated by the unsatisfying results from the previous experiments, we turned to investigate novel methods to partition the data into K experts. This time we wish to partition the labels instead of the images in order to maximize the ccd (Equation 3.7). In this chapter we examine three methods for labels partitioning. The first one is a modification to K-means algorithm, the second method make a use of stacked auto encoders for the partition and the last is a novel ANN architecture for constrained clustering.

4.1 Chunklets K-means

4.1.1 Method

Inspired by [35], for each category we averaged the embedding vectors of its images and applied Kmeans++ on the mean embeddings. As in 3.1, we performed multiple repetitions, and chose the clustering which minimizes the silhouette score of the **original** data points (not the mean embedding).

4.1.2 Experiment Results

We repeated experiment 3.3 and applied Chunklets K-means to feature vectors that were obtained from SAE models (with 0-9 AE layers), to create partitions of the iNaturalist Aves database. According to each partition we trained experts, a gate model and in a few cases also ME models. As we expected, the performances of the experts improved significantly, comparing to previous experiment (mean expert accuracy increased from 76-77% to 85-88%). However, due to the artificial constraints we forced on the clustering algorithm it became more challenging for the gating network to learn the partition, and its accuracy dropped from 93-97% to 40-60%. As a result, the gated ME models performed poorly. See more results in Table 4.1.

Table 4.1: Partitioning with Chunklets K-means

Embedding Dimension	Mean experts Accuracy	Gate Accuracy	Gated ME	
			initial	final
2048	87.6	50.0	74.0	75.2
1024	87.8	48.4	73.0	-
512	88.3	37.5	72.0	-
128	88.1	45.7	73.3	-
64	88.2	44.1	73.1	-
32	87.3	54.0	74.1	-
16	87.2	54.1	73.8	-
8	86.5	61.6	73.5	-
4	85.0	67.1	73.6	-
2	84.7	61.8	72.8	72.0

Table 4.1 Accuracy of the experts (top-5), gate (top-1) and gated ME (top-5), which derived from Chunklets K-means on embedding of variable dimension.

4.2 Maximal Activation Partitioning

We thought we should create a constrained partition that takes into account the ability of the gate model. Motivated by this thought, we designed a special gate model that can **tell us** which partition would make more sense to it. The big question here is how to make such a model do what we want it to do, and the answer is not simple. We tried many designs and training approaches, and the following paragraphs present the method that led to the most satisfying results.

4.2.1 Methods

Architecture Let N denote the number of categories and let K denote the number of wanted experts. We designed a Maximal Activation Partition (MAP) model (Figure 4.1) that ends with an encoding layer of size K followed by a decoding layer of size N. The idea is to train this model for classification and then to use the output of the encoder for the partitioning of the categories according to the maximal activation of the neurons, assuming the encoder acts like some kind of a gate model.

Partitioning How exactly to infer the partition? Before we continue let us make some more notations:

- $S = \{(x_i, y_i)\}_{i=1}^{n} \subset X \times Y$ is the training set, $Y = [N]$.
- $c_X : X \to [K], c_Y : Y \to [K]$ denotes the subset assignment of an image and a label respectively to a subset in the partition, such that for all $k \in [K], y \in [N]$ we have:

  (4.1) $$c_Y(y) = k \Leftrightarrow c_X(x_i) = k \forall i : y_i = y$$

  c_X, c_Y are unknown.

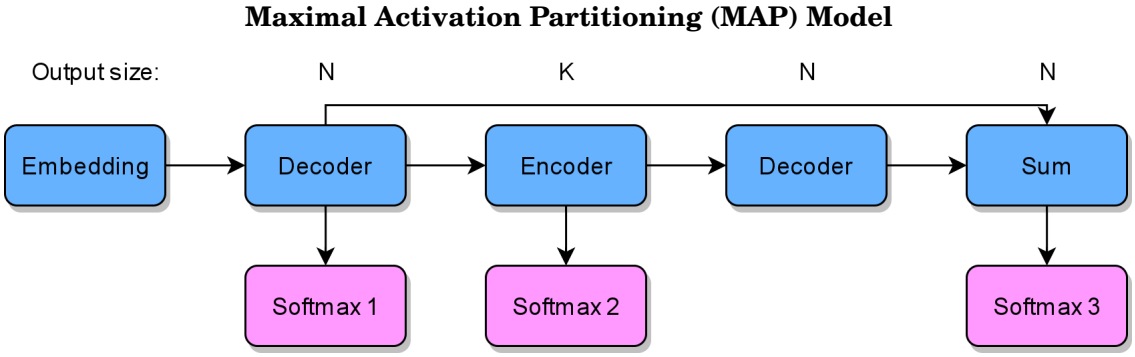


Figure 4.1: MAP has three outputs: Two for category prediction (*softmax1,softmax3*) and the middle one defines the partition (*softmax2*). The first decoder is a fully connected layer. The encoder composed of an ELU activation followed by a fully connected layer and batch normalization. The second decoder composed of an ELU activation followed by a fully connected layer.

- $AE_\theta : X \to [0,1]^K$ denotes the output of the auto encoder as a function of an image (*softmax 2* in Figure 4.1). θ are the parameters of the graph. We look at it as a soft assignment of the images into subsets. i.e. for $x \in X, k \in [K]$:

$$AE_\theta(x)[k] = P(c_X(x) = k) \tag{4.2}$$

Given a trained model, we "know" the soft partition of the images and we want to infer the most likely hard partition of the categories. From Equation (4.1) for some category y and a subset k, the likelihood of $c_Y(y) = k$ is the joint likelihood of $c_X(x) = k$ for all image x labeled with y. Then from (4.2), the most likely subset assignment for y is:

$$\hat{c_Y}(y) \leftarrow argmax_{k \in [K]} \prod_{i:y_i = y} AE_\theta(x_i)[k] \tag{4.3}$$

Unlike to previous experiments, here we made a partition with overlap. Each category was assigned to two experts instead of one in order to increase the number of training images for each expert.

Initialization Suppose our database has a known semantic hierarchy, that is an artificial partition of the categories into super-categories. We could train a ME according to this partition, but such hierarchies are not always visual. Instead, we can leverage the known hierarchy to make an initial partition in the large partitions space, and then fine tune it to a more visual partition in a semi-supervised manner. Inspired by [18], to capture the hierarchy we set the weights of the decoder and encoder to be $W \in \mathbb{R}^{K \times N}$ and W^T respectively where for a category $l \in [N]$ and a super-category $k \in [K]$:

$$W_{i,j} = \begin{cases} 1 & \text{if } h(j) = i \\ -1 & e.w \end{cases}$$

Where $h : [N] \to [K]$ is the known hierarchy.

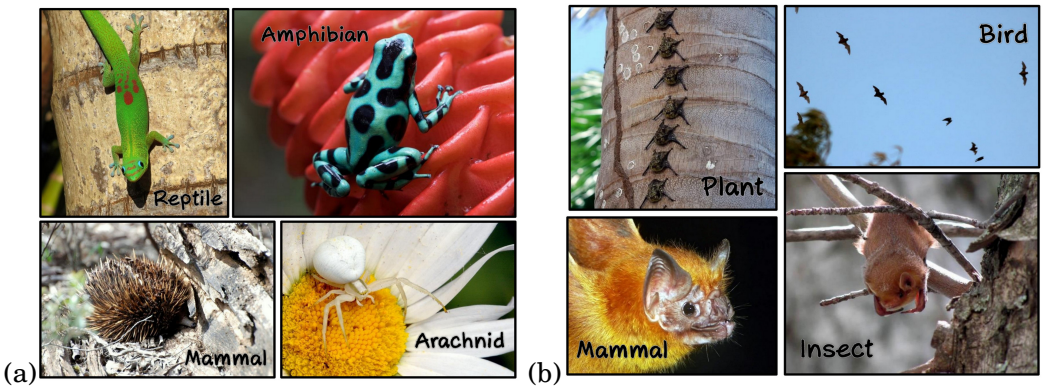
Training We trained this model in two phases: firstly we trained it to learn the known hierarchy, and then we fine tuned it for classification only, leaving the auto encoder layer unsupervised, hoping that it will allow the model to learn a partition which is more visual then the original. Let $\hat{y}, \hat{s}$ be the category and super-category labels respectively of an image x, then the loss function we optimized in the first phase is:

$$H(\hat{y}, softmax1) + H(\hat{s}, softmax2) + H(\hat{y}, softmax3) \tag{4.4}$$

Where H is the cross entropy loss function. The loss in the second training phase is simply $H(\hat{y}, softmax3)$. We used SGD optimizer with momentum and decaying learning rate. In the second phase we set the initial learning rate to be the same as in the middle epoch of the first phase.

4.2.2 Experiment and Results

This experiments was performed on the complete iNaturalist database. We fine tuned Inception v3 from pre-trained ImageNet weights. We then concatenated a MAP architecture to the output of the base model, initialized and trained as described above with $N = 5089, K = 13$. After convergence we derived a partition following the rule of Equation 4.3. The resulted partition was based on the original hierarchy of the species, with minor changes; species moved from one subset to another according to visual properties, as expected. Figure 4.2 presents some examples. According to this partition we trained 13 experts and a gate model, and combined them to a gated ME. The last layers of the combined ME model (all last fully connected layers of the experts and gating network) were trained together with Adam optimizer until convergence. The final model outperformed the base model by almost 6% (Table 4.2). To assess the contribution of the semi supervised partitioning given by the MAP model, we compared it to a ME in which the partition is simply the categories' hierarchy, and indeed we have found that the MAP partition is preferable. Impressive as it may seem, the state of the art results on iNaturalist has less than 5% top-5 error rate [43].



(a) (b)

Figure 4.2: Examples for the data partition obtained by MAP. (a) Species from different super-classes that moved to the plants subset. (b) Bat species that moved to different subsets.

Table 4.2: MAP ME performances

Model	Top-5 Accuracy (%)
Base Model	78.7
Hierarchy ME	82.7
MAP ME	**84.4**

Table 4.2 Comparing performances of the base model and ME models that their partition was driven from MAP model (MAP ME) and the original hierarchy (Hierarchy ME).

4.3 Chunklets Clustering Neural Network (CCNN)

Now let us take the previous idea one step farther. We expected the MAP model to form a partition of the data in such a way that all images of the same category will be mapped into the same subset, but we never said that explicitly. This section describes an architecture that together with its loss function defines a novel constrained clustering algorithm.

4.3.1 Methods

Architecture Let us add another notation to the notations in 4.2.1: Let $C \in \mathbb{R}^{N \times K}$ be a **soft** subset assignment of the categories. So for all $k \in [K], y \in [N]$ we have:

$$C[y,k] = P(c_Y(y) = k) \tag{4.5}$$

Our architecture (Figure 4.3) is composed of:

1. Stacked auto encoders followed by a softmax activation, applied to the embedding of an input image. The structure and training method of the stacked auto encoders are exactly as defined in 3.3.1. The output of the softmax represents AE_θ (defined in 4.2.1). After training the model, we use AE_θ as a gate model, to predict the correct subset (cluster) of an input image.
2. A trainable matrix (a parameter of the graph) which represents C from (4.5), and to its rows we apply softmax activation. After the training the model, we use C to assign the categories to subsets by applying argmax to its rows.

Loss Our goal is to find C, θ such that:

1. The assignment is not soft, i.e. $C[y]$ is approximately a hot-one vector, for all $y \in [N]$.
2. The assignment of an image is the same as the assignment of its category, i.e. for all $(x,y) \in S$, $AE_\theta(x) \approx C[y]$ or at least $argmax_k AE_\theta(x)[k] = argmax_k C[y,k]$.

Considering the above two constraints, the most natural loss for a given $(x,y) \in S$ is:

$$L(x,y;\theta,C) = H(C[y], AE_\theta(x)) = \underbrace{H(C[y])}_{1} + \underbrace{D_{KL}(C[y] \| AE_\theta(x))}_{2} \tag{4.6}$$

Architecture of Chunklets Clustering Neural Network (CCNN)

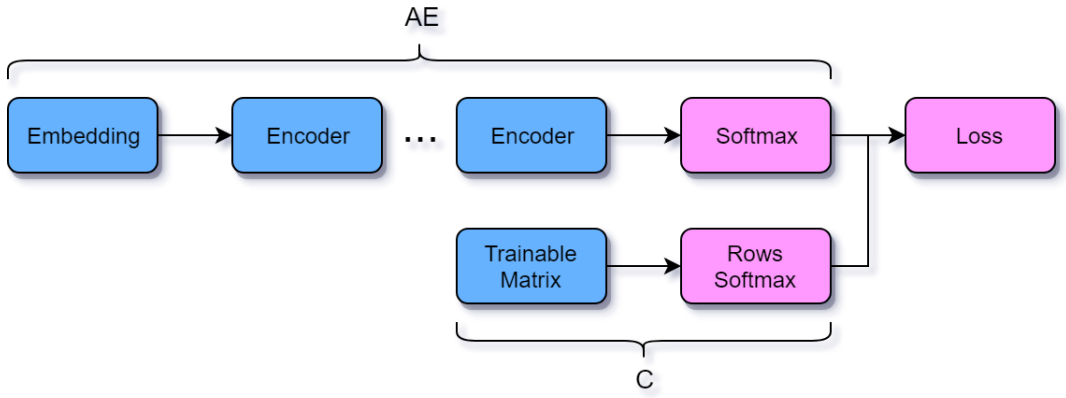


Figure 4.3: CCNN composed of soft subset assignment of the categories (C) and cluster prediction for the input images (AE).

Regularization Notice that a global minimum of the loss function can be achieved if for all $(x,y) \in S$:

$$AE_\theta(x)[k] = C[y,k] = \begin{cases} 1 & \text{if } k = k' \\ 0 & o/w \end{cases}$$

For some k', and we wish to avoid such cases. Moreover, we want the partition to be somewhat balanced since if one expert will have many categories compared to other experts, its training will be a computational bottleneck. If we refer to C as a hard assignment, then $\sum_{y\in[N]} C[y,k]$ is the number of categories assigned to the k'th subset. A perfect balance would be $\sum_{y\in[N]} C[y,k] \approx \frac{N}{K} \Leftrightarrow \frac{1}{N}\sum_{y\in[N]} C[y,k] \approx \frac{1}{K}$ for all $k \in [K]$. In other words, the entropy of the vector $\frac{1}{N}\sum_{y\in[N]} C[y]$ is maximized. Therefore, we want to minimize $-H\left(\frac{1}{N}\sum_{y\in[N]} C[y]\right)$. Altogether, the loss function with the balance regularization is:

$$L(S;\theta,C) = \sum_{(x,y)\in S} H(C[y], AE_\theta(x)) - \gamma H\left(\frac{1}{N}\sum_{y\in[N]} C[y]\right) \underbrace{=}_{Appendix\ A} \sum_{(x,y)\in S} H(C[y], h(x;\theta,C)) \tag{4.7}$$

Where:

$$h(x;\theta,C)[k] = \frac{AE_\theta(x)[k]}{\left(\frac{1}{N}\sum_{y\in[N]} C[y,k]\right)^{\frac{\gamma}{|S|}}} \tag{4.8}$$

Let us analyze the regularized loss function with respect to γ. Consider two cases:

- $\gamma = 0 \Rightarrow h(x;\theta,C) = AE_\theta(x) \Rightarrow L(x,y;\theta,C) = H(C[y], AE_\theta(x))$, as expected.
- $\gamma = |S| \Rightarrow h(x;\theta,C)[k] = \frac{AE_\theta(x)[k]}{\frac{1}{N}\sum_{y'\in[N]} C[y',k]}$. The denominator normalizes the soft assignment of x, by dividing its affinity to the k'th subset ($AE_\theta(x)[k]$) by the average category affinity to the k'th subset ($\frac{1}{N}\sum_{y\in[N]} C[y,k]$).

Training CCNN is trained in a semi supervised way. We do not need any prior knowledge on the hierarchy of the categories, but we feed the optimization algorithm with the true category labels of the images. The parameters of the stacked auto encoders are layer-wise pre-trained for classification, as described in 3.3.1. The initialization of C is more tricky. So far, we initialized

it randomly. However, uniform initialization would maybe fit better or some other initial soft partition.

4.3.2 Experiments and Results

Artificial Data To begin with, we used a miniature CCNN model to cluster a simple artificial data. We sampled 500 points from 4 Gaussian distributions. Each point was labeled 1-4 according to its Gaussian, and we wanted to cluster the labels to two clusters. There are two ways to cluster them, horizontally or vertically, while the horizontal partition is more natural (see Figure 4.4). The simple CCNN model we used has $N = 4, K = 2$ and its AE function is defined by:

1. Input layer of size 2.
2. Embedding - Fully connected layer of size 2, followed by an ELU activation.
3. Prediction - Fully connected layer of size 2, followed by a softmax activation.

We initialized the weights randomly and ran SGD for 6 epochs while keeping track of the cluster prediction of the AE (first row in Figure 4.4) and the embedding of the points (second row in Figure 4.4). We noticed a few important observations. Most importantly, the model behaves as expected and converges efficiently. It is very interesting to see the embedding that the model created for the points in order to give the correct cluster prediction. The data is compressed into one line on the plane even though it is not necessary for a linear separator. Lastly, in most repetitions (65% of 110), the model converged to the horizontal clustering. Figure 4.5 shows some examples for bad clustering. It seems that our algorithm is strongly affected by initialization, more about it in the next paragraph.

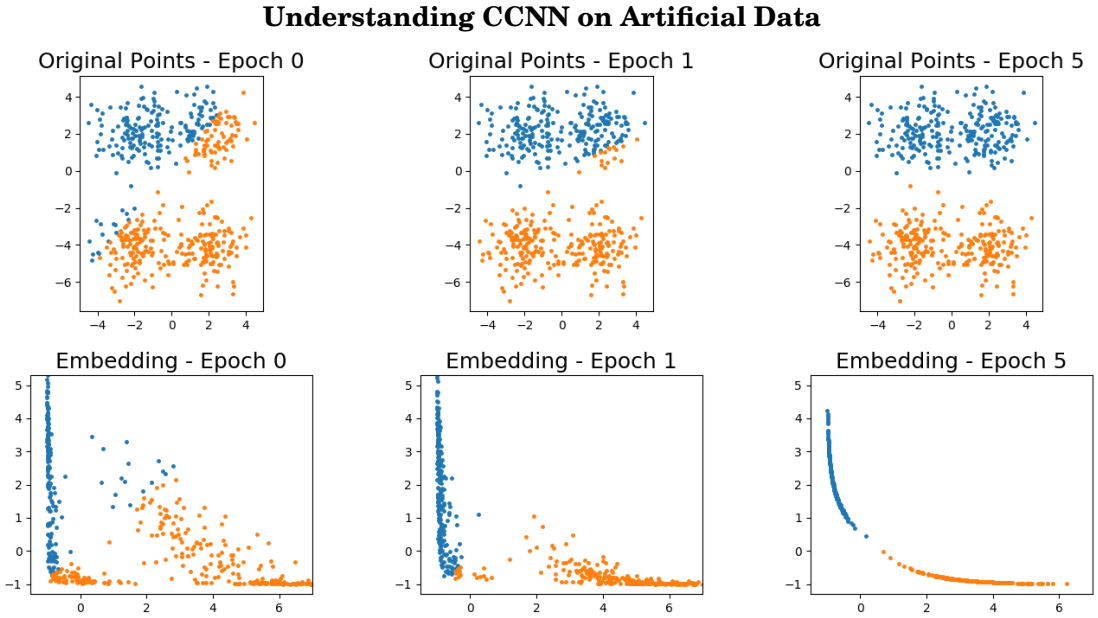


Figure 4.4: The artificial data points (first row), colored according to their cluster prediction that was made by the AE function, and their embedding (second row), along the learning phases.

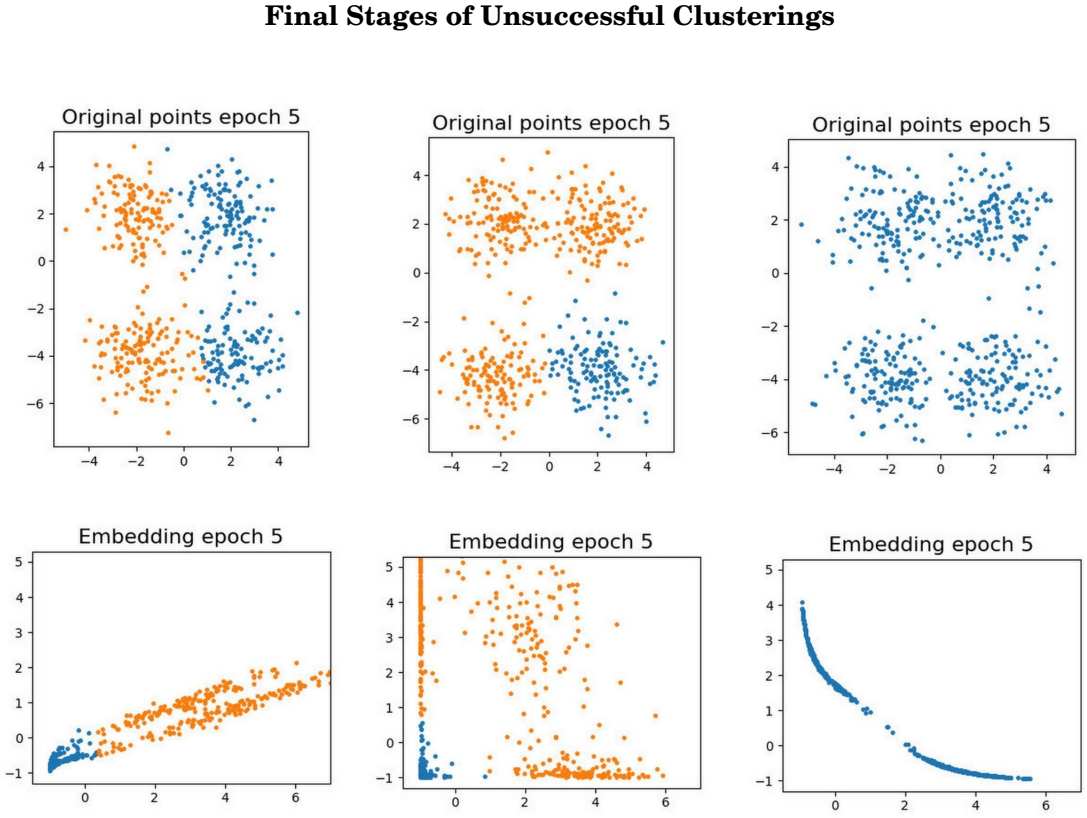


Figure 4.5: Examples of common bad clustering. Original points above, embedding below.

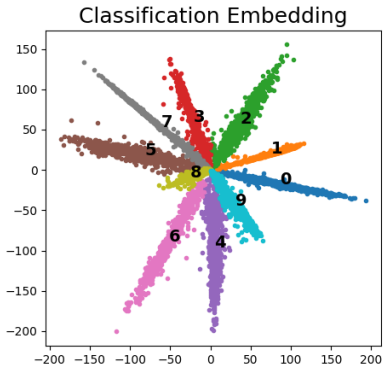


Figure 4.6: The embedding of the digits after classification training.

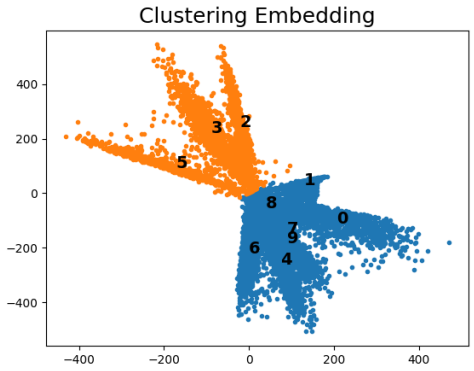


Figure 4.7: The embedding of the digits after CCNN training.

MNIST We trained a simple multilayer perceptron on MNIST classification task. The model has 5 fully connected layers followed by ReLU activation function and Dropout, except for the last layer that has softmax activation. In the first test we used embedding of size 2, in order to visualize it. Figure 4.6 shows the 2D embedding of the model for the classified digit images. After convergence, we trained CCNN with $K = 2$ for 5 epochs to cluster the digits. As in previous experiment, it is very interesting to observe the embedding after the convergence of the CCNN (Figure 4.7).

Since the order of the digits in the embedding space is flexible, it seems that the clustering is

different at every repetition. To measure the stability of our clustering method we repeated this process 100 times and computed for each pair of digits, the probability of them being together in the same cluster (Figure 4.8). We notice that while some of the digits are very likely to be in the same cluster, such as (4,9) and (7,9), about half of the pairs have almost random probability (0.45 to 0.6) of sharing the same cluster. We conclude that our constrained clustering method is very dependent on initialization, like many other clustering methods.

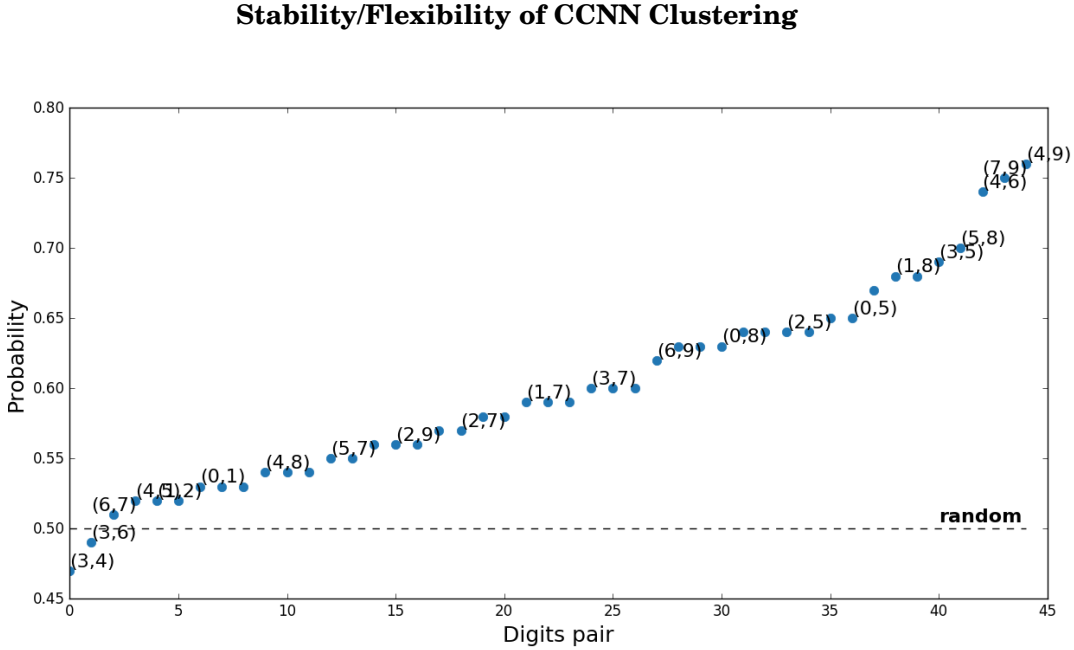


Figure 4.8: The probability of i, j being together in the same cluster, for all $i \neq$j, sorted. (Notice that the labels do not represent all pairs, the points however do)

CIFAR 100 We trained a convolutional neural network with 8 convolution blocks on CIFAR-100 database. We then added to the embedding layer (of size 512) a CCNN model with $N = 100, K = 5$ exactly as described in 4.3, trained it and inferred a partition. Unlike the experiments on artificial data and MNIST, when we trained CCNN on CIFAR with balance regularization factor $\gamma = 0$, the model always converged to the global minimum where one cluster has N categories and the rest are empty. As a consequence, the model predicts the correct cluster for 100% of the validation images. To examine the affect of the balance regularization we repeated the experiment 20 times for each γ in the range [0,80] with skips of 5. We measured the imbalance of a partition by computing standard deviation (std) of the clusters' sizes. Std 0 is a perfect balance and 40 is one full cluster and the rest are empty. For each γ in the tested range we averaged the standard deviation of the clusters' sizes and the accuracies of the models in predicting the correct cluster for the validation set. First thing we noticed in Figure 4.9 is that the balance regularization works as expected. The bigger the γ - the more balanced the partition gets. But unfortunately there is always a price, the bigger the γ - the less accurate the model gets. Figure 4.11 shows clearly the negative correlation between the balance of the partition and the ability to predict the correct cluster.

Affects of the regularization parameter on the balance and the accuracy

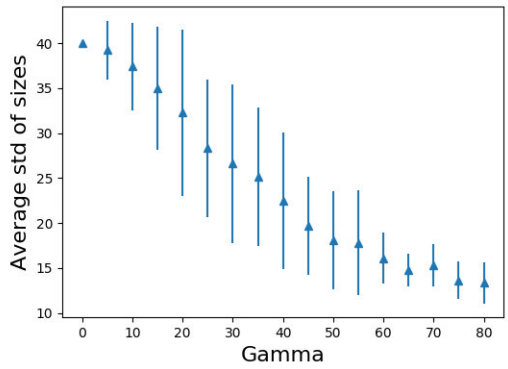


Figure 4.9: Bigger γ reduces the standard deviation of the clusters' sizes, i.e. increases the balance, as expected.

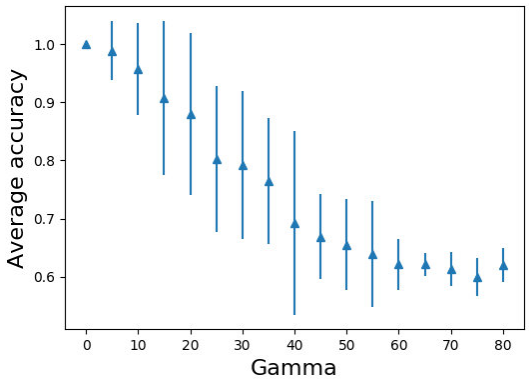


Figure 4.10: Increasing γ decreases the accuracy of the model. Unfortunate but not surprising.

Balance-Accuracy Trade-off

Figure 4.11: A scatter of all experiments on CIFAR (16 γ values, 20 repetitions per each) shows a negative correlation between the accuracy of the model and the balance of the partition it creates.

Tweets Recently many studies attempt to cluster users of social networks [27, 31, 36, 49], mostly of Twitter, due to the availability of its data. Usually it is done by a complex user representation based on its tweets, connections and interests. We addressed the same problem but with a different approach: instead of clustering users, we can cluster tweets under the chunklet constraints - tweets of the same user must be in the same cluster. This way we treat a user as a set of points instead of a single point. Using Twitter streaming API we downloaded all tweets from April 2015 and

randomly chose 100 users that have 200-300 English tweets (eliminating retweets). Tweets were pre-processed and converted to vectors representations using the publicly available implementation of Tweet2Vec [8]. The tweets of each user were divided to train and test set, and the train set was clustered with CCNN with one and two layers, without balance regularization. For comparison, we clustered the tweets also with Chunklets K-means algorithm (4.1) using $K = 2, ..., 10$. For the clusterings of the Chunklets K-means we trained one layer perceptron to learn the new data partition. The resulting clusters were evaluated with the following metrics:

1. Accuracy of the trained model on predicting the correct cluster, given a new tweet.
2. Balance of the clustering by computing the entropy of the clusters' sizes, divided by N, the number of users (See Regularization in 4.3.1).
3. Split Accuracy: in another test we splitted the training tweets of each user into to two "new" users and measured the percentage of splitted users that were assigned to the same cluster as expected.

Figures 4.12-4.15 show controversial results. While CCNN is better at accuracy, i.e. it creates more separable partitions, Chunklets K-means creates more balanced partitions and it performs better in the split test.

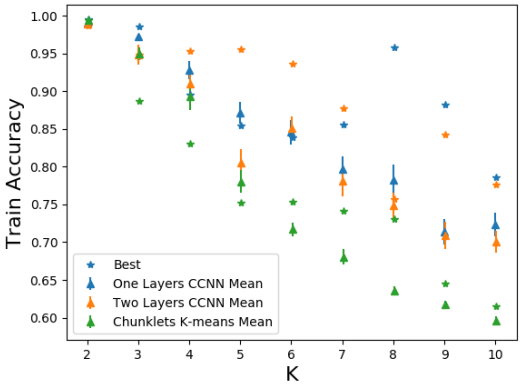


Figure 4.12: Predicting the correct clusters for tweets in the training set.

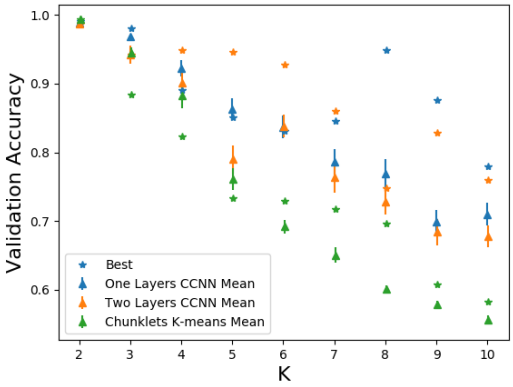


Figure 4.13: Predicting the correct clusters for tweets in the test set.

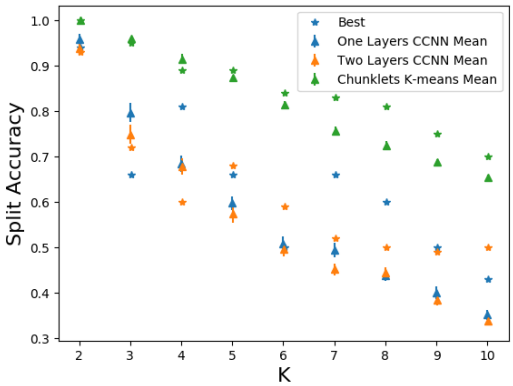


Figure 4.14: Percentage of splitted users that were assigned to the same cluster as expected.

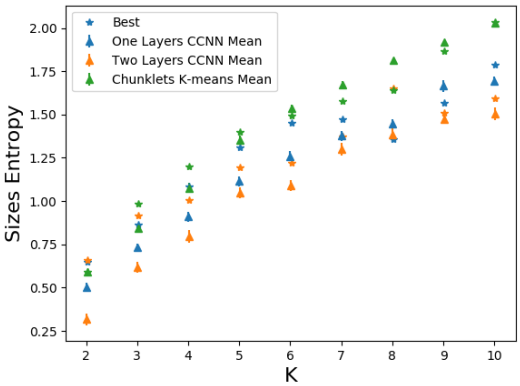


Figure 4.15: Entropy of the clusters' sizes, divided by N.

Figures 4.12-4.15: Performances of CCNN and Chunklets K-means on Tweets. Triangles mark the mean and standard error of 25 experiments for each algorithm. Stars represent the best result, which was chosen by maximizing the geometric average of the train accuracy and the balance.

Chapter 5

Summary and Discussion

We compared various gating methods for combining a mixture of CNNs for image classification, and suggested ways of combining more than one gating method while alternating between fusion and selection. In this series of experiments the partition of the data to experts was first made by applying K-means to the embedding of the images. Later we suggested a dimension reduction using stacked auto encoders and applied K-means on the embedding in lower dimension. After spending thousands of GPU hours the results were mostly inconclusive. Designing a complex module that will outperform the baseline model is apparently not trivial, especially when your baseline model is Inception v3. The way I see it, an unsuccessful ME should be explained considering its two forms: with and without end-to-end training. Prior to training the whole model end-to-end, there is no over-fit problem, as long as the experts do not over-fit. However, this is a very delicate model that has to be very accurate to outperform the baseline model: the experts must be better on their subsets of data and the gating function must be able to predict the correct expert with high probability. We could not find this perfect balance, except for 4.2. On the other hand, when applying an end-to-end training, the partition is much more dynamic: experts on one domains can contribute to other domains, if they know relevant features and if the gating network is flexible enough. In this case the ME model is in danger of over-fit, as number of trainable parameters can be very large (about 120M in our experiments in Chapter 3). [33] argue and prove that in some cases an end-to-end training may require an exponential number of examples, compared to the modular approach. When we repeated the whole experiment in 3.3 with GoogLeNet we still achieved inconclusive results. When we applied the algorithm in 4.2 on CIFAR we noticed an extreme over-fit even with $K = 2$.

Nonetheless, one certain conclusion that turned out from the experiments in Chapter 3 is that if one prefers to avoid an expensive end-to-end training, the gated ME is the best choice. We hypothesized that partitioning the categories instead of the images may improve the performances of the experts and therefore will improve the modular ME (without end-to-end training). We suggested Chunklets K-means as a baseline algorithm for category partitioning algorithm, and repeated the experiments in Chapter 3. While the experts indeed showed significant performance

improvement, the gating network was more challenged, and as a result the combined ME performed poorly. We suggested the MAP model that leverage a known semantic hierarchy of the data, and tested it on the complete iNaturalist database. We saw how it applies unsupervised minor changes to the hierarchy based partition to form a more visual and more natural partition. MAP based ME showed an improvement comparing to the baseline model and to the hierarchy based ME. This model should be investigated more. For example, the contribution of the overlap of the subsets (each category was assigned to two experts) is unclear. MAP model can generate a partition of the images instead of categories, this should be investigated as well.

Finally, we suggested the novel CCNN architecture that clusters labeled data by mapping the data points and the labels to soft clustering, and minimizing the cross-entropy between the mapping of the labels and the mapping of their respective data points. We examined CCNN's basic features by applying it to simple databases. We saw that CCNN is sensitive to initialization, like many other clustering algorithms, but we can run it multiple times and choose the instance that minimizes the loss function. We suggested a regularization function to control the balance of the clusters and demonstrated with it the balance-accuracy trade-off. Lastly, we cluster Twitter users by optimizing a CCNN on their tweets. The resulting CCNN can predict the correct cluster of a new tweet with much better accuracy than Cunklets K-means, but it performs less well in the split test. To conclude, CCNN model may be found very usefull in clustering Chunklets of data, for example in social networks, but it needs some fine tuning. Farther work on this model should include smarter initialization, more regularization functions and different optimization algorithms.

APPENDIX A

CCNN Loss Analysis

In the following two pages you can an analysis of the regularized loss function of CCNN as described in 4.3.

Claim:

Let:

- $\{(x_i, y_i)\}_{i=1}^{n} \subset X \times Y$. Y is a finite set. And suppose that $|\{i: y_i = y\}| = \frac{n}{N}$ for all $y \in Y$.
- $f: X \to [0,1]^K$
- $g: Y \to [0,1]^K$
- $\lambda \in \mathbb{R}_{>0}$

Then:

$$\sum_{i=1}^{n}\left[H\big(g(y_i), f(x_i)\big)\right] - \lambda H\left(\frac{1}{N}\sum_{y\in Y} g(y)\right) = \sum_{i=1}^{n} H\big(g(y_i), h(x_i)\big)$$

Where:

$$h(x)[k] = \frac{f(x)[k]}{\left(\frac{1}{N}\sum_{y\in Y} g(y)[k]\right)^{\frac{\lambda}{n}}}$$

Proof:

Denote $\overline{g(k)} = \frac{1}{N}\sum_{y\in Y} g(y)[k]$

$$\begin{aligned}
&\sum_{i=1}^{n}\left[H\big(g(y_i), f(x_i)\big)\right] - \lambda H\left(\frac{1}{N}\sum_{y\in Y} g(y)\right) \\
&\qquad = \left(\sum_{i=1}^{n}\left[H\big(g(y_i), f(x_i)\big)\right]\right) + \left(\sum_{k=1}^{K}\left[\left(\sum_{y\in Y}\frac{\lambda}{N} g(y)[k]\right)\log\big(\overline{g(k)}\big)\right]\right) \\
&\qquad = \left(\sum_{y\in Y}\sum_{i:y_i=y} H\big(g(y), f(x_i)\big)\right) + \left(\sum_{k=1}^{K}\sum_{y\in Y}\left[\frac{\lambda}{N} g(y)[k]\log\big(\overline{g(k)}\big)\right]\right) \\
&\qquad = \left(\sum_{y\in Y}\sum_{i:y_i=y} H\big(g(y), f(x_i)\big)\right) + \left(\sum_{y\in Y}\sum_{k=1}^{K}\left[\frac{\lambda}{N} g(y)[k]\log\big(\overline{g(k)}\big)\right]\right) \\
&\qquad = \sum_{y\in Y}\left[\left(\sum_{i:y_i=y} H\big(g(y), f(x_i)\big)\right) + \sum_{k=1}^{K}\left[\frac{\lambda}{N} g(y)[k]\log\big(\overline{g(k)}\big)\right]\right]
\end{aligned}$$

$$= \sum_{y \in Y} \sum_{i:y_i=y} \left[H\big(g(y), f(x_i)\big) + \frac{N}{n} \left(\sum_{k=1}^{K} \left[\frac{\lambda}{N} g(y)[k] \log\big(\overline{g(k)}\big) \right] \right) \right]$$

$$= \sum_{y \in Y} \sum_{i:y_i=y} \left[-\sum_{k=1}^{K} [g(y)[k] \log f(x_i)[k]] + \sum_{k=1}^{K} \left[\frac{\lambda}{n} g(y)[k] \log\big(\overline{g(k)}\big) \right] \right]$$

$$= \sum_{y \in Y} \sum_{i:y_i=y} -\sum_{k=1}^{K} \left[g(y)[k] \log f(x_i)[k] - \frac{\lambda}{n} g(y)[k] \log\big(\overline{g(k)}\big) \right]$$

$$= \sum_{y \in Y} \sum_{i:y_i=y} -\sum_{k=1}^{K} \left[g(y)[k] \left(\log f(x_i)[k] - \frac{\lambda}{n} \log\big(\overline{g(k)}\big) \right) \right]$$

$$= \sum_{y \in Y} \sum_{i:y_i=y} -\sum_{k=1}^{K} \left[g(y)[k] \log \frac{f(x_i)[k]}{\overline{g(k)}^{\frac{\lambda}{n}}} \right] = \sum_{i=1}^{n} -\sum_{k=1}^{K} \left[g(y_i)[k] \log \frac{f(x_i)[k]}{\overline{g(k)}^{\frac{\lambda}{n}}} \right]$$

$$= \sum_{i=1}^{n} -\sum_{k=1}^{K} [g(y_i)[k](\log h(x_i)[k])] = \sum_{i=1}^{n} H\big(g(y_i), h(x_i)\big)$$

Where $h: X \to \mathbb{R}^K_{>0}$ s.t $h(x)[k] = \frac{f(x)[k]}{\overline{g(k)}^{\frac{\lambda}{n}}}$

BIBLIOGRAPHY

[1] C. C. AGGARWAL, *On k-anonymity and the curse of dimensionality*, in Proceedings of the 31st international conference on Very large data bases, VLDB Endowment, 2005, pp. 901–909.

[2] D. ARTHUR AND S. VASSILVITSKII, *k-means++: The advantages of careful seeding*, in Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[3] S. BASU, A. BANERJEE, AND R. MOONEY, *Semi-supervised clustering by seeding*, in In Proceedings of 19th International Conference on Machine Learning (ICML-2002, Citeseer, 2002.

[4] T. BERG, J. LIU, S. WOO LEE, M. L. ALEXANDER, D. W. JACOBS, AND P. N. BELHUMEUR, *Birdsnap: Large-scale fine-grained visual categorization of birds*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2011–2018.

[5] M. BILENKO, S. BASU, AND R. J. MOONEY, *Integrating constraints and metric learning in semi-supervised clustering*, in Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 11.

[6] L. BREIMAN, *Bagging predictors*, Machine learning, 24 (1996), pp. 123–140.

[7] A. DEMIRIZ, K. P. BENNETT, AND M. J. EMBRECHTS, *Semi-supervised clustering using genetic algorithms*, Artificial neural networks in engineering (ANNIE-99), (1999), pp. 809–814.

[8] B. DHINGRA, Z. ZHOU, D. FITZPATRICK, M. MUEHL, AND W. W. COHEN, *Tweet2vec: Character-based distributed representations for social media*, arXiv preprint arXiv:1605.03481, (2016).

[9] Z. GE, A. BEWLEY, C. MCCOOL, P. CORKE, B. UPCROFT, AND C. SANDERSON, *Fine-grained classification via mixture of deep convolutional neural networks*, in Applications of Computer Vision (WACV), 2016 IEEE Winter Conference on, IEEE, 2016, pp. 1–6.

[10] Z. GE, C. MCCOOL, C. SANDERSON, AND P. CORKE, *Modelling local deep convolutional neural network features to improve fine-grained image classification*, in Image Processing (ICIP), 2015 IEEE International Conference on, IEEE, 2015, pp. 4112–4116.

[11] Z. GE, C. MCCOOL, C. SANDERSON, AND P. CORKE, *Subset feature learning for fine-grained category classification*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2015, pp. 46–52.

[12] H. GOËAU, P. BONNET, AND A. JOLY, *Plant identification in an open-world (lifeclef 2016)*, in CLEF 2016-Conference and Labs of the Evaluation forum, 2016, pp. 428–439.

[13] J. GOODBAND, O. C. HAAS, AND J. A. MILLS, *A mixture of experts committee machine to design compensators for intensity modulated radiation therapy*, Pattern recognition, 39 (2006), pp. 1704–1714.

[14] S. GROSS, M. RANZATO, AND A. SZLAM, *Hard mixtures of experts for large scale weakly supervised vision*, arXiv preprint arXiv:1704.06363, (2017).

[15] S. GUTTA, J. R. HUANG, P. JONATHON, AND H. WECHSLER, *Mixture of experts for classification of gender, ethnic origin, and pose of human faces*, IEEE Transactions on neural networks, 11 (2000), pp. 948–960.

[16] J. V. HANSEN, *Combining predictors: comparison of five meta machine learning methods*, Information Sciences, 119 (1999), pp. 91–105.

[17] T. HERTZ, A. BAR-HILLEL, AND D. WEINSHALL, *Boosting margin based distance functions for clustering*, in Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, p. 50.

[18] H. HU, G.-T. ZHOU, Z. DENG, Z. LIAO, AND G. MORI, *Learning structured inference neural networks with label relations*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2960–2968.

[19] R. A. JACOBS, *Bias/variance analyses of mixtures-of-experts architectures*, Neural computation, 9 (1997), pp. 369–383.

[20] R. A. JACOBS, M. I. JORDAN, AND A. G. BARTO, *Task decomposition through competition in a modular connectionist architecture: The what and where vision tasks*, Cognitive science, 15 (1991), pp. 219–250.

[21] R. A. JACOBS, M. I. JORDAN, S. J. NOWLAN, AND G. E. HINTON, *Adaptive mixtures of local experts*, Neural computation, 3 (1991), pp. 79–87.

[22] D. KLEIN, S. D. KAMVAR, AND C. D. MANNING, *From instance-level constraints to space-level constraints: Making the most of prior knowledge in data clustering*, tech. rep., Stanford, 2002.

[23] T. KOHONEN, *The self-organizing map*, Proceedings of the IEEE, 78 (1990), pp. 1464–1480.

[24] S. KOTSIANTIS, *Combining bagging, boosting, rotation forest and random subspace methods*, Artificial Intelligence Review, 35 (2011), pp. 223–240.

[25] S. B. KOTSIANTIS, *An incremental ensemble of classifiers*, Artificial Intelligence Review, 36 (2011), pp. 249–266.

[26] Z. LI, J. LIU, AND X. TANG, *Constrained clustering via spectral regularization*, in Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 421–428.

[27] S. LIANG, E. YILMAZ, AND E. KANOULAS, *Dynamic clustering of streaming short documents*, in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2016, pp. 995–1004.

[28] S. MASOUDNIA AND R. EBRAHIMPOUR, *Mixture of experts: a literature survey*, Artificial Intelligence Review, 42 (2014), pp. 275–293.

[29] C. MCCOOL, Z. GE, AND P. I. CORKE, *Feature learning via mixtures of dcnns for fine-grained plant classification.*, in CLEF (Working Notes), 2016, pp. 511–517.

[30] C. MCCOOL, T. PEREZ, AND B. UPCROFT, *Mixtures of lightweight deep convolutional neural networks: applied to agricultural robotics*, IEEE Robotics and Automation Letters, 2 (2017), pp. 1344–1351.

[31] Z. QIU AND H. SHEN, *User clustering in a dynamic social network topic model for short text streams*, Information Sciences, 414 (2017), pp. 102–116.

[32] R. E. SCHAPIRE, *The strength of weak learnability*, Machine learning, 5 (1990), pp. 197–227.

[33] S. SHALEV-SHWARTZ AND A. SHASHUA, *On the sample complexity of end-to-end training vs. semantic abstraction training*, arXiv preprint arXiv:1604.06915, (2016).

[34] A. J. SHARKEY AND N. E. SHARKEY, *Combining diverse neural nets*, The Knowledge Engineering Review, 12 (1997), pp. 231–247.

[35] N. SHENTAL, A. BAR-HILLEL, T. HERTZ, AND D. WEINSHALL, *Computing gaussian mixture models with em using equivalence constraints*, in Advances in neural information processing systems, 2004, pp. 465–472.

[36] K. SINGH, H. K. SHAKYA, AND B. BISWAS, *Clustering of people in social network based on textual similarity*, Perspectives in Science, 8 (2016), pp. 570–573.

[37] C. SZEGEDY, V. VANHOUCKE, S. IOFFE, J. SHLENS, AND Z. WOJNA, *Rethinking the inception architecture for computer vision*, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[38] B. TANG, M. I. HEYWOOD, AND M. SHEPHERD, *Input partitioning to mixture of experts*, in Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on, vol. 1, IEEE, 2002, pp. 227–232.

[39] B. THOMEE, D. A. SHAMMA, G. FRIEDLAND, B. ELIZALDE, K. NI, D. POLAND, D. BORTH, AND L.-J. LI, *Yfcc100m: The new data in multimedia research*, Communications of the ACM, 59 (2016), pp. 64–73.

[40] T. P. TRAN, T. T. S. NGUYEN, P. TSAI, AND X. KONG, *Bspnn: boosted subspace probabilistic neural network for email security*, Artificial Intelligence Review, 35 (2011), pp. 369–382.

[41] K. TUMER AND J. GHOSH, *Error correlation and error reduction in ensemble classifiers*, Connection science, 8 (1996), pp. 385–404.

[42] E. D. ÜBEYLI, K. ILBAY, G. ILBAY, D. SAHIN, AND G. AKANSEL, *Differentiation of two subtypes of adult hydrocephalus by mixture of experts*, Journal of medical systems, 34 (2010), pp. 281–290.

[43] O. S. Y. C. Y. S. C. S. A. A. H. P. P. B. S. VAN HORN, GRANT; MAC AODHA, *The inaturalist species classification and detection dataset*, arXiv eprint arXiv:1707.06642, (2017).

[44] P. VINCENT, H. LAROCHELLE, I. LAJOIE, Y. BENGIO, AND P.-A. MANZAGOL, *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion*, Journal of Machine Learning Research, 11 (2010), pp. 3371–3408.

[45] K. WAGSTAFF AND C. CARDIE, *Clustering with instance-level constraints*, AAAI/IAAI, 1097 (2000), pp. 577–584.

[46] K. WAGSTAFF, C. CARDIE, S. ROGERS, S. SCHRÖDL, ET AL., *Constrained k-means clustering with background knowledge*, in ICML, vol. 1, 2001, pp. 577–584.

[47] J. XIE, R. GIRSHICK, AND A. FARHADI, *Unsupervised deep embedding for clustering analysis*, in International conference on machine learning, 2016, pp. 478–487.

[48] S. E. YUKSEL, J. N. WILSON, AND P. D. GADER, *Twenty years of mixture of experts*, IEEE transactions on neural networks and learning systems, 23 (2012), pp. 1177–1193.

[49] Y. ZHAO, S. LIANG, Z. REN, J. MA, E. YILMAZ, AND M. DE RIJKE, *Explainable user clustering in short text streams*, in Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, ACM, 2016, pp. 155–164.